M. Tech (C.S) Dissertation Report

Learning of Conoidal Structures in Connectionist Framework

A dissertation submitted towards partial fulfillment of the requirements for the **M.Tech. (Computer Science)** degree of Indian Statistical Institute

by
Anirban Das

Under the supervision of

Dr. Jayanta Basak — Machine Intelligence Unit
Prof. S.K. Pal — Machine Intelligence Unit

INDIAN STATISTICAL INSTITUTE
203, Barrackpore Trunk Road
Calcutta-700035

Certificate of Approval

This is to certify that the thesis entitled *Learning of Conoidal Structures in Connectionist Framework* submitted by *Anirban Das*, towards partial fulfillment of the requirement for *M. Tech.* in *Computer Science* degree of the *Indian Statistical Institute, Calcutta*, is an acceptable work for the award of the degree.

Date : July 23, 2000

(Supervisor)

(External Examiner)

Acknowledgement

My sincerest gratitude goes to *Dr. J.Basak & Prof. S.K.Pal* for their guidance, advice, enthusiasm and criticisms throughout the course of this dissertation.

I would also like to take this opportunity to thank *Dr. Palash Sarkar, Dr. D.P. Mukherjee & Dr. Subhomoy Maitra* for their excellent courses which helped me in this work.

Last but not the least, I would like to thank *Anil Ghosh and Kishori Mohon Konwar* for their valuable advises.

Author

Anirban Das

Abstract

A two layer neural network model is designed which accepts image coordinates as the input and learns the parametric form of conoidal shapes (lines/circles/ellipses) adaptively. It provides an efficient representation of visual information embedded in the connection weights and the parameter of the processing elements. It not only reduces the large space requirements as in classical Hough transform, but also represents parameters with high precision, even in presence of noise. The performance of the methodology is compared with other existing algorithms and has been found to excel over those algorithms in many cases.

Contents

Chapter 1

Introduction

Hough transform [4,5], developed by Paul Hough (1962) [4], is a method to detect parameterized models (e.g. curves and features) in data by mapping data features into manifolds in the parameter space. The parameters, that are consistent with many of the manifolds correspond to curves in the image and thus methods that find peaks in the parameter space can be used to detect the image curves.

It has a wide range of applications in medical sciences (e.g. detection of chest ribs [30], breast cancer cells [31]), in molecular biology (e.g. detection of shapes of microbes [32]), in image processing (e.g. detection of edges [33], corner points, 3-D shape [34], in segmentation techniques [35], detection of motion parameters [36]), in automatic guided vehicles (detection of roads, islands at the cross points [37]), in satellite picture processing (e.g. time domain analysis of solar coronal structures [38], target detection and tracking [39]), in data analysis (e.g. estimation of regression parameters, detection of echelle orders [40,41]), in pattern recognition (e.g. automatic recognition of Hebrew letters [42], ridge counting in fingerprints [43]), in geography (e.g. matching subgraphs derived from geographical maps [44]), in data compression (e.g. contour or waveform data compression [45]) and many other fields.

1.0.1 How Hough Transform works

Let us explain the way of computing classical Hough transform for lines, circles etc. Note that, if a curve in the image space is continuous and second order differentiable, then the points lying on the curve will lie in a contiguous space after Hough transformation. Let a shape be represented parametrically as $f(x, y, \Theta) = 0$ where $\Theta = [\theta_1, \theta_2, ..., \theta_n]$ is a parameter vector. For example, a straight line can be represented as

$$x \cos \phi + y \sin \phi - r = 0 \qquad (1.1)$$

Here $\Theta = [\phi, r]$. Similarly a circle can be represented as

$$(x - c_1)^2 + (y - c_2)^2 - r^2 = 0 \tag{1.2}$$

where (c_1, c_2) is center and r is the radius. Here $\Theta = [c_1, c_2, r]$. In similar way an ellipse can be parameterized as

$$ax^2 + by^2 + 2gx + 2fy + c = 0 \tag{1.3}$$

Here $\Theta = [a, b, g, f, c]$. Given an image with set of object pixels $\mathbf{X} = [x_1, x_2, ..., x_N]$, the task is to find out the parametric shapes depicted by the pixels in the image by mapping the object pixels into parameter space. In order to do that an accumulator array $A(\Theta)$ is maintained. For each object pixel (x_i, y_i) all the parameter space is searched to find out Θ such that $f(x_i, y_i, \Theta) = 0$ The accumulator array A is updated using equation $A(\Theta) \leftarrow A(\Theta) + c$, where c is the contribution present in each slot corresponding to a particular curve segment by that pixel. Thus an object pixel in the image space is transformed to the parameter space such that, the pixel votes for a particular parametric curve in the parameter space. Thus for a subset of object pixel forming a parametric shape in image space all the cumulative parametric curves will pass a single point Θ' in the parametric space. This Θ' is a unique parameter vector, which lie on intersection of all parametric curves passing through the object pixel. Actually, object pixels are clustered around local peaks in the accumulator. The task is to identify this peaks in the parameter space. Algorithmically, [algorithm A.0.1]
Step 1: Create a set of coordinates from the object pixels in the image.
Step 2: Transform each coordinate in (x, y) into a parameterized curve in the parameter space.
Step 3: Increment the cells in the parametric space determined by the parametric curve.
Step 4: Detect local maxima in the accumulator array. Each local maximum may correspond to parametric curves in the image space.
Step 5: Extract the curve segments using using the knowledge of maximum positions.

1.0.2 Advantages and disadvantages of Hough Transform

Advantages

Since Hough transform is based on accumulating evidence of the presence of the shape in the image space, it works fine even if broken, deformed or fragmented segments are present in the image. In order to detect lines or curves present in the image, continuity is not essential. It works gracefully in case of occlusion. The method is very robust to the addition of random data produced by poor image segmentation. In addition to this, the contribution $A(\Theta) = A(\Theta) + c$ (step 3 of algorithm CHT) can be modified to incorporate gray values also where c represents contribution depending on the gray value. Thus Hough transform can be applied on gray images directly, without going through any prior image segmentation.

Again, each object pixel is treated independently, therefore the method can be implemented using more than one processing unit, i.e. parallel processing of all points is possible.

Disadvantages

Peak detection in the accumulator space is one of the major problem of Hough transform. In order to identify the peaks suitably, selection of a suitable threshold is necessary. However the threshold may depend on image quality (presence of noise in it). Beside this, the method requires large amount of computational time and space. Computational time is proportional to the size of the image and grows exponentially with no. of elements in the parameter vector. Accuracy of finding the parameters is limited by the resolution of accumulator array.

1.0.3 What and why Neural Networks

The neural networks are massively parallel interconnection of simple processing elements called neurons, intended to interact with environment in a way the biological nervous systems do. Artificial neural networks (ANN) or simply neural nets, are also known as connectionist models, parallel distributed processing models, neuromorph models. Different applications of ANN include speech and image recognition (which includes surface interpolation, edge detection, shape from shading, velocity field estimation, color determination, structure from motion [26]), optimization problems (e.g. travelling salesman problem [20]), dynamic robot motion planning [18], various control system schemes [19].

The simplest node sums N weighted inputs and passes the result through a nonlinearity (e.g. hard limiters, threshold logic elements and sigmoidal nonlinearities). The node is characterised by an internal threshold or offset θ. Neural network models are specified by the network topology node characteristic and training and learning rules. This rules specify initial set of weights and indicate how weights should be adaptive during use to improve performance.

The potential benefits of neural networks are : first, they are capable of learning situations adaptively after being trained on a number of examples

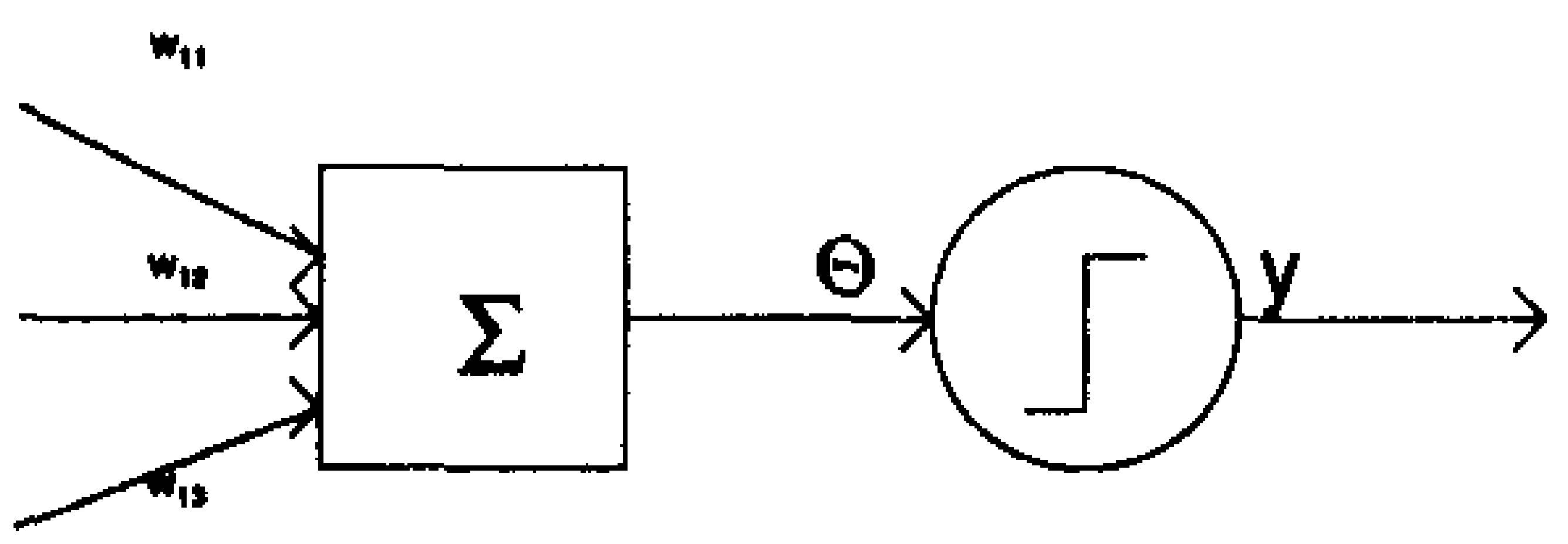

Figure 1.1: Simplest Neural Network

of a relationship, as the network designer chooses the weights a priori or just random. In the second case the network performs desired computation by iterative adjustments of weight strengths by (a) supervised learning, where learning is done on the basis of direct comparison of the output of the network with known correct answers and (b) unsupervised learning, where only available information is in the correlation of the input data or signals. Second, their ability to generalize to new situations since, after being trained on a number of examples of a relationship, they can often induce a complete relationship that interpolates and extrapolates from the examples in a sensible way. The third is their high computation rates provided by massive parallelism, since activation value in each node can be changed independently. The fourth is their fault tolerance, since, if some spurious activation exist in the parameter space, they would loss in competition in associating the pixels and would not get further support from those pixels.

Their main drawback is the long training times required particularly for problems of large dimensionality. This is in contrast to the human vision system which is capable of learning from a small number of examples. Another problem with ANNs is the lack of any theoretical guidance concerning the choice of number of nodes, activation functions etc.

1.0.4 Contributions in the dissertations

A neural network model is designed to identify straight line, circle, ellipse segments present in the image (or hyper-planes, hyper-spheres, hyper-ellipsoids in multidimensional inputs). A two layer neural network model is proposed which accepts image coordinates as the input and learns the parametric form of conoidal shapes (lines/circles/ellipses) in the image (or planes/ spheres/ellipsoids/ hyper-planes/hyper-spheres/hyper-ellipsoids in higher dimensional models) adaptively.

The learning rule of Hough transform network is extended to accept conoidal shapes in addition to straight line segments. Also a new way of reducing localization error has been devised. The problem of detection of various parametric shaped segments is classified according to the level of difficulty, and a few typical results are presented. The outputs are then compared with different existing algorithms including classical Hough transform(CHT), randomised Hough transform(RHT), adaptive fuzzy c-shell clustering algorithm with unconstrained shape & orientation matrix (AFCS-U). The network is again extended to learn parametric shapes in gray level images .

1.0.5 Summary

In this chapter the classical Hough transform is described and its problems are mentioned. The relevance of attacking the problem of determining parametric shapes using neural network model is argued, and possible advantage and disadvantages are stated. Finally the present contributions in the dissertation are presented and its advantage over the other models are mentioned.

In the next chapter the existing neural network implementation of Hough transform and other variations of the transform is described.

Chapter 2

Literature Survey

2.1 Hough transform Network to detect straight lines

A two layer neural network model viz. Hough transform network has been proposed [1,2] to detect straight lines in image (or planes/hyper-planes in higher dimension).

Network Architecture

The network consists of two layers, viz. input and output layer. Each input node is connected to all output nodes. The number of input nodes is equal to the dimensionality of input space. For example the network consists of only two input nodes when it accepts the image coordinates in the object region as the input. The number of output nodes is chosen to be equal to the number of straight line segments in the image. The connection weights from the input layer to any particular node along with the threshold of the output node represent the parameter values of the corresponding straight line segments in the image or the hyper-plane on the case of higher dimensional input. The network accepts image coordinates sequentially as the input and learns adaptively the parameter values in the form of connection weights in an unsupervised mode. The activation in the output layer indicates the straight line segments (or hyper-plane) to which the input pixel (or vector) belongs.

Philosophy behind the Network

A straight line can be parametrically expressed as

$$x_1 \cos\phi + x_2 \sin\phi = r \tag{2.1}$$

where $\mathbf{x} = (x_1,x_2)$ is the coordinate of any point on the straight line and (r,ϕ) are the parameters of the straight line. Eqn. (2.1) can be represented as

$$\sum_i w_i x_i = \Theta \tag{2.2}$$

where $\mathbf{x} = (x_1,x_2)$ is the coordinate of a pixel in the object region of the image. Corresponding to x, the input to the neural network, the output vector $\mathbf{y}$ is given as

$$\mathbf{y} = f(\mathbf{u}) = [f(u_1), f(u_2), ..., f(u_m)] \tag{2.3}$$

where $\mathbf{u}$ is the input to the output neurons, given as

$$\mathbf{u} = \mathbf{W}\mathbf{x} - \Theta \tag{2.4}$$

$f(.)$ is a bell shaped transfer function with peak at zero. A Gaussian form of the function is chosen i.e.

$$y_i = exp(-u_i^2/\lambda^2) \tag{2.5}$$

If x belongs to the i^{th} hyper-plane then $u_i = 0$ and $y_i = 1$, otherwise $y_i < 1$. To learn $\mathbf{W}$ and Θ the objective function (a continuous and differentiable one)

$$E = [\prod_{i=1}^{n}(1 - y_i)]^{\alpha} \tag{2.6}$$

is chosen, such that it is zero when y_i is zero for any i. The parameter $\alpha > 0$ determines the steepness of the function near minima(i.e. a fixed point). As α increases, the steepness decreases. The weight matrices $\mathbf{W}$ and Θ are updated in order to decrease E.

Learning Rules

Following the steepest gradient descent rule the weight updating or the learning rules for $\mathbf{W}$ and Θ are

$$\Delta w_{ij}(t) = -\frac{u_i(t)(\frac{y_i(t)}{1-y_i(t)})[x_j(t) - w_{ij}(t)(u_i(t) + \theta_i(t))]}{2\alpha\sum_k(\frac{y_k(t)}{1-y_k(t)})^2 \log(\frac{1}{y_k(t)})[k + X - (u_k(t) + \theta_k(t))^2]} \tag{2.7}$$

$$\Delta \theta_i(t) = -\frac{u_i(t)(\frac{y_i(t)}{1-y_i(t)})}{2\alpha\sum_k(\frac{y_k(t)}{1-y_k(t)})^2 \log(\frac{1}{y_k(t)})[k + X - (u_k(t) + \theta_k(t))^2]} \tag{2.8}$$

The updating rules reveal the fact that, the changes in the parameter values are independent of the selection of λ. The parameter λ is however implicitly embedded in the output of the network. Initially, when the parameter values are far from the fixed point, the denominator is very small. To account for this situation, the learning rate is clamped to a constant value when the denominator is very small. Near the optimal solution, the learning rate changes according to the equation (2.7) and (2.8).

2.2 Variations of Hough Transform

Two methods that have been widely used are mapping sets of image pixels into the parameter space and randomization.

Shapiro [22] first considered a variation of Hough transform where the edge points are mapped into all of the curves in the parameter space that satisfy the error model for the edge point. Shapiro's application of these ideas was to use the accumulator method, with the modification that all of the accumulator cells consistent with the error model for a particular point receive votes. This process has extreme computational requirements. Shapiro suggested the use of large grid cells to reduce this problem, despite the inherent loss of resolution and curve discrimination performance.

Stephens [43] formulated a variant of the Hough transform in terms of maximum likelihood estimation. A probability density function for the features is used that has a uniform component modeling the pixels that are not on the curve and a component that falls off as a Gaussian with the distance from the curve to model the pixels that are on the curve. This method yields correct propagation of localization error in terms of a Gaussian error distribution, but it is computationally expensive.

Breuel [46] described a line detection technique related to the Hough transform that searches hierarchical subdivisions of the parameter space using a bounded error model and thus avoids some of the problems of the accumulator method. In this technique, the parameter space is divided into cells that are tested to determine whether they can contain a line that passes within the bounded localization error of a specified number of pixels. If the cell cannot be ruled out, the cell is divided and the procedure is repeated recursively. This continues until the cells become sufficiently small, at which point they are considered to be lines satisfying the output criterion.

Murakami et al. [15] described algorithms for performing straight line detection using a one-dimensional accumulator. One of the algorithms maps pairs of feature points into the one-dimensional accumulator by taking the angle of the line between the points. The algorithm is structured in such a way that only the pairs containing one of the feature points are examined in each iteration. This can be viewed as a precursor to the decomposition techniques that we describe. Unfortunately, this algorithm had $O(n^2)$ complexity, where n is the number of image features and did not consider the effects of localization error.

Xu & Oja [7] described the randomized Hough transform (RHT) where, the interest was to reduce computation time of Hough transform by randomly choosing a subset of object pixels in the image. For the detection of curves with N parameters, they map sets of N image pixels into single points in the parameter space and accumulate votes in a discretized version of parameter space. The pixel sets that are mapped into the parameter space are chosen randomly and the votes are accumulated until a sufficient peak is found or some threshold number of sets have been examined. They also described a robust stopping criterion for this procedure and they approximated the time required by the algorithm by modeling it as a generalized Bernoulli process.

Liang [14] discussed a Hough transform technique where sets of image pixels are mapped to single points in a discretized parameter space by fitting curves to the pixels in small windows of the image. This method allows a fast implementation and a low storage requirement, but detection performance will degrade in the presence of image noise, due to poor local fits, and in cluttered images, due to distractors present in the image windows.

Bergen and Shvaytser [17] gave a theoretical analysis of the use of randomization techniques to speed up the Hough transform. They considered both mapping individual pixels and mapping sets of pixels into the parameter space. Their method achieved a computational complexity independent of the number of edge pixels in the image, but with two caveats. First, only curves that represent some predetermined fraction of the total number of edge pixels are found. Second, the method is allowed to be in error by a fractional parameter all of the time and by greater than this fractional parameter with some small frequency. The practicality of this method is questionable, since the constant number of random samples that must be examined is often very large. In fact, this number may often be larger than the number of different samples that are possible.

Kiryati [9] et al. used randomization to improve the running time of the standard Hough transform. They simply subsampled the edge pixels in the image and proceeded with the standard algorithm. Their analysis implies that the computational requirements of the Hough transform can be improved, while the performance is degraded little.

Califano and Bolle [47] used a multiple window parameter transform to exploit long distance information in the extraction of parameterized objects from image data. Lateral inhibition is used in a connectionist-like framework to improve the accuracy. In addition, a radius of coherence is defined for each pixel to reduce the computation time required by the method.

Leavers [11] described a technique called the dynamic generalized Hough transform(DGHT), where she used the technique of mapping image pixels into a single point in the parameter space and, furthermore, in each iteration selected a single image pixel to constrain the transform, which must be present in each of the sets of pixels that are mapped into the parameter space.

C.F. Olson [8] found a similar technique viz. 'Constrained Hough transform', which describes techniques to perform fast and accurate curve detection, where localization error can be propagated efficiently into the parameter space. Here, Hough transform is modified to allow the formal treatment localization error. Hough transform can be subdivided into many small subproblems without a decrease in performance, where each subproblem is constrained to consider only those curves that pass through some subset of the edge pixels up to the localization error. This property allows to accurately and efficiently propagate localization error into the parameter space such that curves are detected robustly without finding false positives. The use of randomization techniques yields an algorithm with a worst-case complexity of $O(n)$, where n is the number of edge pixels in the image, if we are only required to find curves that are significant with respect to the complexity of the image.

Leavers [12] used a storage efficient voting mechanism in a discretized parameter space, where the votes are projected onto each of the parameter space axes and several one-dimensional accumulators are kept. While this method of accumulating votes reduces that amount of memory that is required, it may exacerbate problem of false alarms if the votes in the parameter space are not sparse. Olson [8] found an additional technique that has proven in the efficient implementation of Hough transform techniques is a multi resolution or coarse-to-fine search of the parameter space to find peaks. For example, Li, Lavin & Master [14] recursively divided the parameter space in hyper-cubes in a coarse-to-fine hierarchy. At each level of the hierarchy, only those hyper-cubes that receive enough votes to surpass some threshold are passed on to the next level for examination.

Galambos et al. [23] presented a variation of Hough transform algorithm referred to as Progressive probabilistic Hough transform (PPHT). Unlike the probabilistic HT where classical HT is performed on a pre-selected fraction of input points, PPHT minimises the amount of computation needed to detect lines by exploiting the difference in the fraction of votes needed to detect reliably lines with different numbers of supporting points. The fraction of points used for voting need not be specified ad hoc or using a priori knowledge, as in the probabilistic HT; it is a function of the inherent complexity of the input data. The algorithm is ideally suited for real-time applications with a fixed amount of available processing time, since voting and line detection is interleaved.

Kazuhide Sugawara [24] extended the Hough transform to incorporate the notion of weight, and apply it to reducing spurious line detection caused by the presence of large non-text areas with many black pixels. The weighting function is defined according to the proportion of black pixels in the area, giving a low weight to an area with an extremely high proportion of black pixels.

Soodamani and Liu [25] found an efficient method known as fuzzy Hough transform(FHT) of updating fuzzy evidences which enhances recognition of approximate shapes. They found that the global evidence can be found from lócal support through a sliding window, from local statistical variability measure and also by some fuzzy support measures.

Chapter 3

Detection of Conoidal Structures

3.1 Parametric Shapes

Before going into the description of the neural architecture and its learning rules, let us provide here briefly the mathematical form of the parametric shapes to be detected by the network.

Definition 1:The hyper-ellipsoidal shell prototype, S_A at center $\mathbf{v}$ having size r is the set

$$S_A(\mathbf{v}, r, \mathbf{A}) = \{\mathbf{x} \in \Re^p | (\mathbf{x} - \mathbf{v})^{\mathbf{T}} \mathbf{A}_i (\mathbf{x} - \mathbf{v}) = r^2\} \tag{3.1}$$

where $\mathbf{A}$ is a $p \times p$ symmetric positive definite matrix.

Note that, any conic sectional shape can be parametrically represented in generalised 2-dimensional equation as

$$A_{11}(x_1 - v_1)^2 + (A_{12} + A_{21})(x_1 - v_1)(x_2 - v_2) + A_{22}(x_2 - v_2)^2 = r^2 \tag{3.2}$$

If $\mathbf{A}$ is a positive definite symmetric matrix (which means $\mathbf{A} = \mathbf{A}^T$), then the equation will represent an ellipse, where $\mathbf{x} = (x_1, x_2)$ is the coordinate of any point on an ellipse with center at $\mathbf{v} = (v_1, v_2)$.

If we scale down $\mathbf{A}_i$ by r_i, and fix r_i to 1, we get a five parameter representation of an ellipse. In specific case, when $\mathbf{A}$ is an identity matrix, the equation will represent a circle.

A double straight line of equations

$$(x - v_1) + \alpha(y - v_2) = k \tag{3.3}$$

and

$$(x - v_1) + \alpha(y - v_2) = -k \tag{3.4}$$

can be represented jointly as

$$[(x - v_1) + \alpha(y - v_2)]^2 = k^2 \tag{3.5}$$

Comparing the coefficients of the above equation with equation. we get

$$\mathbf{A} = \begin{bmatrix} 1 & \alpha \\ \alpha & \alpha^2 \end{bmatrix} \tag{3.6}$$

Definition 2: The hypershell prototype, S_A, at center $\mathbf{v}(\mathbf{v}\in\Re^p)$ is the set

$$S_{\mathbf{A}}(\mathbf{v}, \mathbf{A}) = \{\mathbf{x}\in\Re^p | (\mathbf{x} - \mathbf{v})^{\mathbf{T}}\mathbf{A}(\mathbf{x} - \mathbf{v}) = 1\} \tag{3.7}$$

where $\mathbf{A}$ is a $p\times p$ symmetric positive definite matrix. This matrix accounts for the size, eccentricity and orientation of the ellipsoid.

Definition 3: Distance D_{ik} of point x_k from the i^{th} hyper-spherical shell prototype $S_I(\mathbf{v}_i, \mathbf{A}_i)$ is defined as

$$(D_{ik})^2 = ([(\mathbf{x}_k - \mathbf{v}_i)^T\mathbf{A}_i(\mathbf{x}_k - \mathbf{v}_i)]^{\frac{1}{2}} - 1)^2 \tag{3.8}$$

Let $\mathbf{x} = (x_1, x_2)$ be the coordinate of a pixel in the object region of the image. In the case of higher dimensional input $\mathbf{x} = (x_1, x_2, ...x_n)$ represents the variable on the hypershell to be identified. The distance D_i of point k^{th} feature point $\mathbf{x}_k$ from the i^{th} hypershell prototype $S_A(\mathbf{v}_i, r_i, \mathbf{A}_i)$ is defined as

$$D_{ik} = [(\mathbf{x}_k - \mathbf{v}_i)^T\mathbf{A}_i(\mathbf{x}_k - \mathbf{v}_i)]^{0.5} - r_i \tag{3.9}$$

where $\mathbf{v}_i$ and r_i are center and radius of the prototype.

3.2 Network Architecture

The proposed network model consists of two layers, viz. input and output layer. Each input node is connected to all output nodes. The number of input nodes is equal to the dimensionality of input space. For example. the network consists of only two input nodes when it accepts the image coordinates of the object pixels. The number of output nodes is chosen to be equal to or more than total number of circle/ellipse segments in the image or total number of hypersphere/hyperellipsoid segments in case of multidimensional input. The network accepts image coordinates sequentially as the input and learns adaptively the parameter values in the form of connection weights and thresholds in an unsupervised mode.

Each output node i stores a vector $\mathbf{v}_i$ representing the center of the parametric shape (e.g. circle or ellipse). The output node also stores a matrix

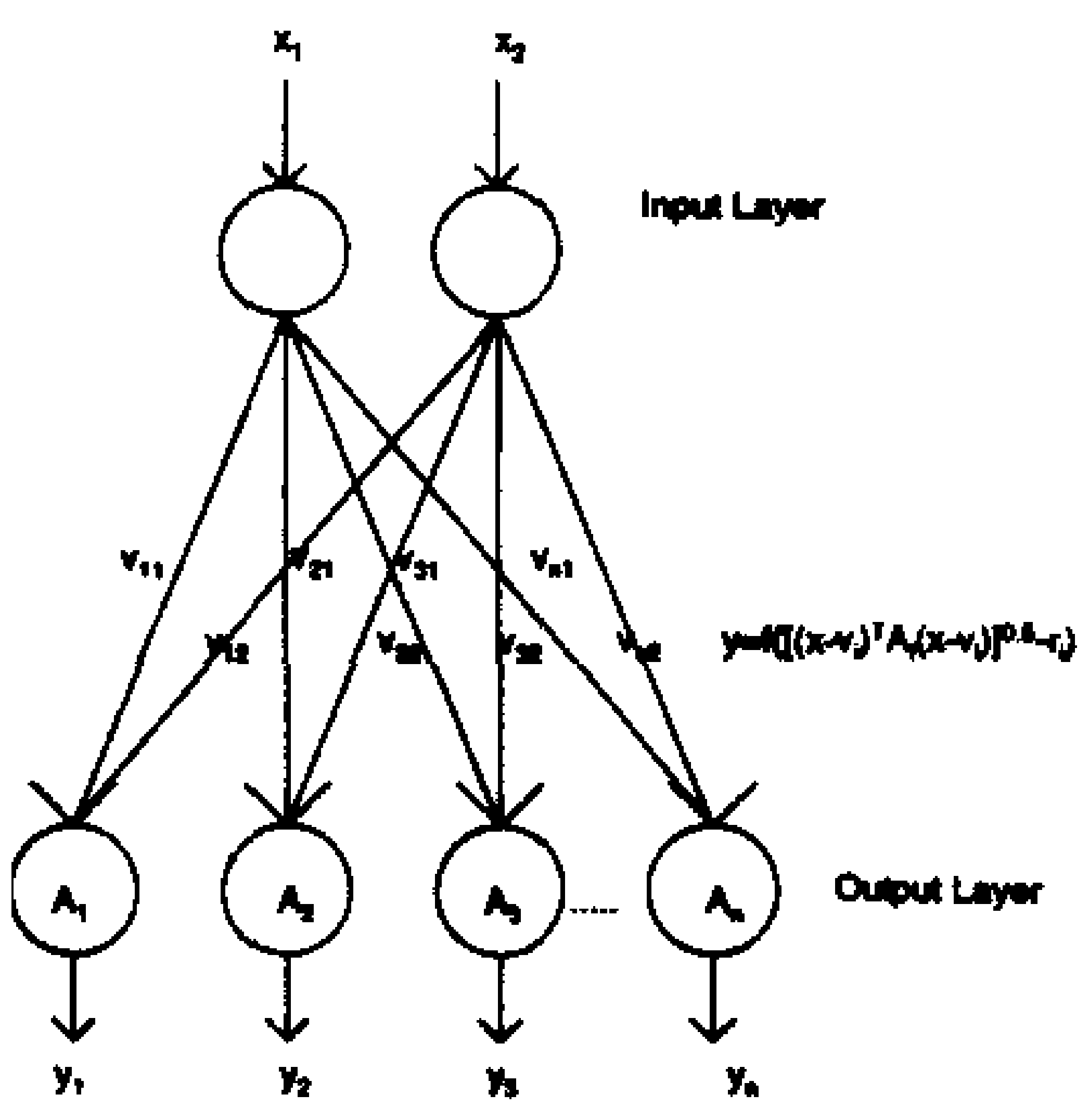

Figure 3.1: Network Architecture

$\mathbf{A}_i$ to reflect the orientation and size of the parametric shape. Note that this kind of architecture is analogous to radial basis function networks (RBF) where the hidden nodes represent certain receptive fields given by certain basis function $\phi_j(\|\mathbf{a} - \mu_j\|/\sigma j)$, $j = 1, 2, ..., J$ and the normalizing factor σ_j decides the range of influence of the j^{th} unit. Here, the k^{th} unit in the output layer of the network is given by $b_k' = \sum_{j=0}^{J} w_{kj} h_j$ where $h_j = \phi_j(.)$ and $h_0(= -1)$ is the output of the bias unit, so that w_{k0} corresponds to the bias on the k^{th} output unit. The nonlinear basis function $\phi_j(.)$ of the j^{th} hidden unit is a function of the normalized radial distance between the input vector $\mathbf{a}=(a_1, a_2, ..., a_M)^T$ and the weight vector $\mu_j=(\mu_{j1}, \mu_{j2}, ..., \mu_{jM})^T$ associated with the unit, μ_j is analogous to $\mathbf{v}_i$, weight vector of the proposed model. However the present model does not provide any supervised information like RBF in the output layer.

The vector centers and the matrix $\mathbf{A}_i$ are adaptively updated in unsupervised manner. The output vector $\mathbf{y} = [y_1, y_2, ..., y_n]$, corresponding to $\mathbf{x}$ is given as $y_i = f([(\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x}_k - \mathbf{v}_i)]^{0.5} - r_i)$ where r_i is a threshold of output node i; i.e. $f(u_i) = [f(u_1)f(u_2)...f(u_m)]$ where u_i is the internal state which correspond to the distance $\mathbf{D}$ in equation (3.9) and m is the number of output nodes. $f(.)$ is an on-centre bell-shaped function. $f(.)$ can be chosen as a Gaussian basis function i.e. $y_i = exp(-u_i^2/\lambda^2)$. Note that $f(.)$ can have other forms also including Cauchy distribution function of the form $y_i = \frac{1}{1+\frac{u_i^2}{\lambda^2}}$ [2]. The parameter λ determines the width of the bell shaped function. For small values of λ the activation function is highly localised. Therefore the hypershell becomes slightly perturbed by the distant points. This is necessary when the dynamics of the network settles to the desired

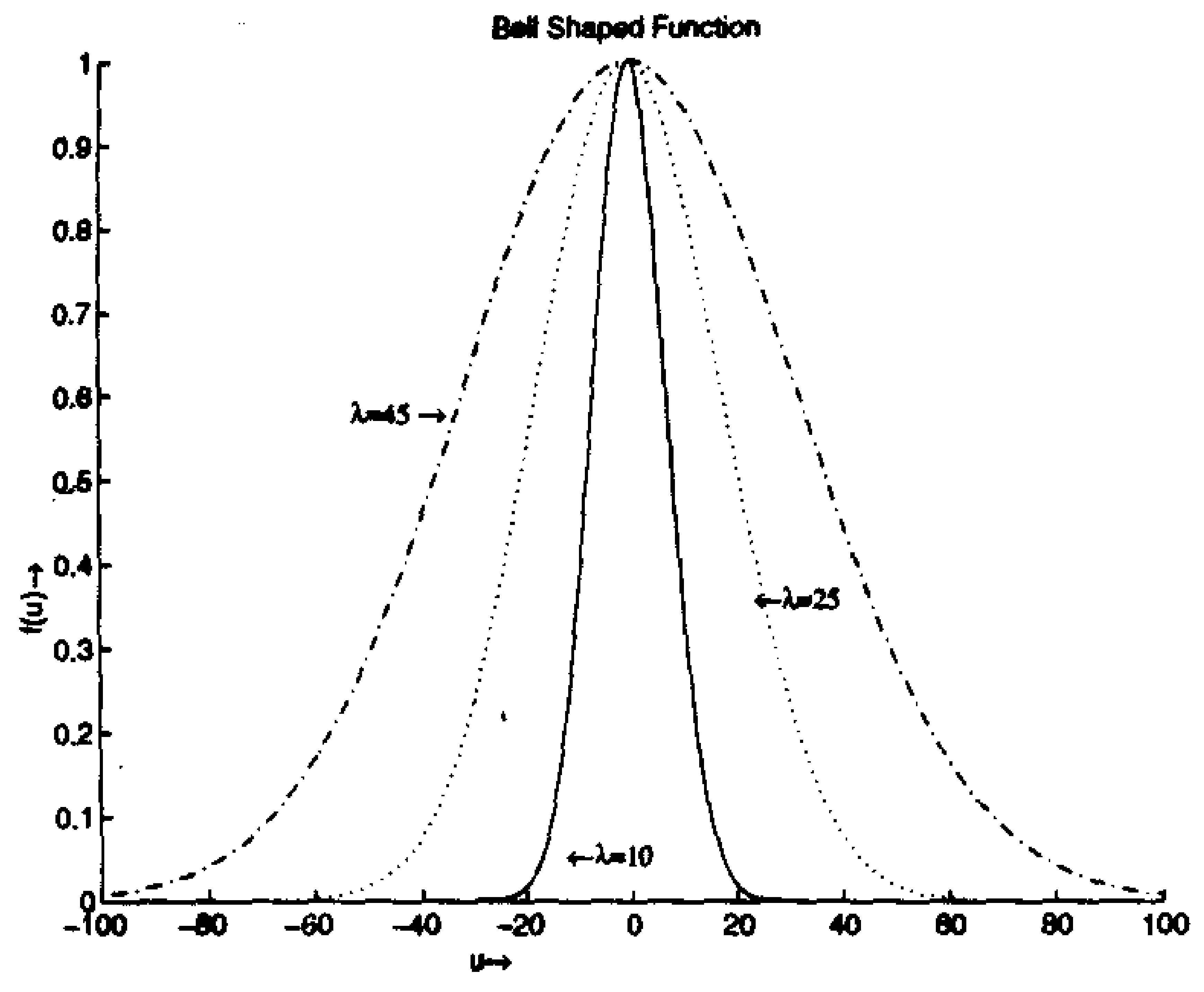

Figure 3.2: Gausian form of function f

stable state. On the other hand for large λ the attraction of the hypershells towards the distant points are high, thereby making the network inefficient.

If $\mathbf{x} = (x_1, x_2)$ belongs to the i^{th} hypershell then $y_{ik} = 1$ otherwise $y_{ik} < 1$. The activation in the output layer indicates the line/circle/ellipse segments (or hyperplane/hypersphere/hyperellipsoid) to which the input pixel (or vector) belongs.

3.3 Learning Rules

To learn $\mathbf{v}$ and r, a continuous and differentiable objective function

$$E = [\prod_{i=1}^{n}(1 - y_i)]^{\alpha} \tag{3.10}$$

is chosen such that it is zero when y_i is 1 for any i. The parameter $\alpha > 0$ determines the steepness of the function near local minima. As α increases, steepness decreases. $\mathbf{v}$ and $\mathbf{A}$ are updated in order to decrease E. Following the steepest gradient descent rule, the weight updating is given as

$$\Delta \mathbf{v}_i = -\beta \frac{\partial E}{\partial \mathbf{v}_i} \tag{3.11}$$

where β is the constant of proportionality. From (3.11), $\Delta\mathbf{v}_i$ can be written as,

$$\Delta\mathbf{v}_i = \frac{\gamma E}{(1-y_i)} f'(u_i) \frac{\partial u_i}{\partial \mathbf{v}_i} \tag{3.12}$$

where, $\gamma = \alpha\beta$ and

$$\frac{\partial u_i}{\partial \mathbf{v}_i} = \frac{1}{2(u_i + r_i)}[-2\mathbf{A}_i(\mathbf{x}-\mathbf{v}_i) + (\mathbf{A}_i - \mathbf{A}_i^T)(\mathbf{x}-\mathbf{v}_i)] \tag{3.13}$$

If $\mathbf{A}_i$ is a symmetric matrix, $\mathbf{A}_i = \mathbf{A}_i^T$. Similarly change in $\mathbf{A}_i$ can be given as

$$\Delta\mathbf{A}_i = \frac{\gamma E}{(1-y_i)} f'(u_i) \frac{\partial u_i}{\partial \mathbf{A}_i} \tag{3.14}$$

When $\mathbf{A}_i$ is a symmetric matrix,

$$\frac{\partial u_i}{\partial \mathbf{A}_i} = \frac{1}{2(u_i + r_i)}[2(\mathbf{x}-\mathbf{v}_i)(\mathbf{x}-\mathbf{v}_i)^T - diag[(\mathbf{x}-\mathbf{v}_i)(\mathbf{x}-\mathbf{v}_i)^T]] \tag{3.15}$$

where $diag(\mathbf{M})$ is a diagonal matrix containing only the diagonal elements of $\mathbf{M}$.

3.4 Selection of Learning Rate

If learning rate γ is very small then the network will converge very slowly, on the other hand if γ is very large, then there will be an oscillation, and hence the network will not converge to desired parametric shapes. Therefore the solution of the optimal learning rate γ is essential. γ in every updating step should be selected in such a way that E goes to a minimum level. If $E(t)$ is present value of E at t then $E(t) + \Delta E(t) = 0$, $\therefore \Delta E(t) = -E(t)$.

To eliminate γ from (3.12), (3.14) we have, change in output y_i due to small change in u_i given by,

$$\Delta y_i = -\frac{2u_i y_i}{\lambda^2}\Delta u_i \tag{3.16}$$

where

$$\Delta u_i = (\frac{\partial u_i}{\partial \mathbf{v}_i})^T \Delta\mathbf{v}_i + trace((\frac{\partial u_i}{\partial \mathbf{A}_i})^T \Delta\mathbf{A}_i) \tag{3.17}$$

Changed objective function E for small change in output of i^{th} node Δy_i can be written as

$$E(t+1) = E(t)[\prod_i (1 - \frac{\Delta y_i}{1-y_i})]^\alpha \tag{3.18}$$

Near convergence, this can be written as

$$E(t+1) = E(t)[1 - \alpha \sum_i \frac{\Delta y_i}{1 - y_i}] \tag{3.19}$$

$E(t+1)$ being optimised right hand side of (3.19) can be equated to zero, hence

$$\sum_{i=1}^{n} \frac{\Delta y_i}{1 - y_i} = \frac{1}{\alpha} \tag{3.20}$$

so, from eqn (3.12), (3.14), (3.17), (3.20), we have

$$\frac{\gamma E}{\lambda^2} = \frac{1}{\alpha \sum_{i=1}^{n} \log(\frac{1}{y_i})(\frac{1}{u_i+r_i})^2(\frac{y_i}{1-y_i})^2[(\frac{\partial P}{\partial \mathbf{v}_i})(\frac{\partial P}{\partial \mathbf{v}_i})^T + trace(W^T W)]} \tag{3.21}$$

where $\mathbf{W} = 2\mathbf{X}\mathbf{X}^T - diag(\mathbf{X}\mathbf{X}^T)$, $\mathbf{X} = (\mathbf{x} - \mathbf{v}_i)$, $P = (\mathbf{x} - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x} - \mathbf{v}_i)$ and $\frac{\partial P}{\partial \mathbf{v}_i} = -2\mathbf{A}_i(\mathbf{x} - \mathbf{v}_i)$. Thus finally learning rules are given by,

$$\Delta \mathbf{v}_i = -\frac{(\frac{y_i}{1-y_i})(\frac{u_i}{u_i+r_i})(-2\mathbf{A}_i\mathbf{X})}{\alpha \sum_{i=1}^{n} \log(\frac{1}{y_i})(\frac{1}{u_i+r_i})^2(\frac{y_i}{1-y_i})^2[(\frac{\partial P}{\partial \mathbf{v}_i})(\frac{\partial P}{\partial \mathbf{v}_i})^T + trace(\mathbf{W}^T\mathbf{W})]} \tag{3.22}$$

$$\Delta \mathbf{A}_i = -\frac{(\frac{y_i}{1-y_i})(\frac{u_i}{u_i+r_i})(2\mathbf{X}\mathbf{X}^T - diag(\mathbf{X}\mathbf{X}^T))}{\alpha \sum_{i=1}^{n} \log(\frac{1}{y_i})(\frac{1}{u_i+r_i})^2(\frac{y_i}{1-y_i})^2[(\frac{\partial P}{\partial \mathbf{v}_i})(\frac{\partial P}{\partial \mathbf{v}_i})^T + trace(\mathbf{W}^T\mathbf{W})]} \tag{3.23}$$

The updating rules reveal that the changes in parameter values are independent of the selection of λ. The parameter λ is however implicitly embedded in the output $\mathbf{y}$ of the network. Initially when the parameter values of the network are far from the fixed point, the denominator is very small. To account for this situation the learning rate is clamped to a constant value when the denominator is very small. Near the optimal solution the learning rate changes according to the eqn.s (3.22) & (3.23) respectively.

Chapter 4

Implementation and Comparison

4.1 Implementation and Experimental Results

It is found that the learning rate of cluster center matrix **v** and cluster orientation and size matrix **A** are not same. Hence, to define the rate of learning of the parameters of the model two multiplication constants are chosen heuristically, in such a way that, the model works most efficiently and the distance definitions are meaningful, that is A_i remains positive definite. Here, while keeping r fixed the two heuristic values chosen are $\frac{c}{\sqrt{K}}$ for matrix **A** and $\sqrt{K}$ for matrix **v** where K is the average distance of feature points from the origin, so that the ratio of two learning rates remain in the order K. The constant c is found to be approximately equal to 0.5.

Note that, the heuristic values provide desirable output for a wide range of image sizes. It is also found experimentally that the network performance is not very sensitive to the selection of c so long as it is close to 0.5. Further investigations can be made to justify why these heuristic values provide good performance.

The straight lines appearing in an image is considered as a double straight line will be represented as in equation (3.5). Thus one extra line will appear in an image, which may not be present at all. However this is considered as a similar case of detecting a partial circular or elliptic shapes and can be eliminated, checking limit of the shapes depicted in the image. This discussion is out of the scope here.

Detection of circular shapes can be viewed as a special case of detection of elliptic shapes where **A** is taken as identity matrix. A circle can be represented as

$$(x - v_1)^2 + (y - v_2)^2 = r^2 \tag{4.1}$$

where $\mathbf{v} = (v_1, v_2)$ denotes the cluster center and r denotes the cluster radius. So to detect a circular shape it is sufficient to learn only three parameters viz. (v_1, v_2, r). For a circular model $S_A(\mathbf{v}_i, r_i)$ distance from point x to i^{th} hypershell is given as

$$u_i = [(\mathbf{x} - \mathbf{v}_i)^T(\mathbf{x} - \mathbf{v}_i)]^{0.5} - r_i \tag{4.2}$$

So the learning rules will become

$$\Delta \mathbf{v}_i = -\beta\frac{\partial E}{\partial \mathbf{v}_i} = -\beta\frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial u_i}\frac{\partial u_i}{\partial \mathbf{v}_i} \tag{4.3}$$

$$\Delta r_i = -\beta\frac{\partial E}{\partial r_i} = -\beta\frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial u_i}\frac{\partial u_i}{\partial r_i} \tag{4.4}$$

To eliminate β we use eqn. (3.19) and

$$\Delta u_i = (\frac{\partial u_i}{\partial \mathbf{v}_i})^T \Delta \mathbf{v}_i + (\frac{\partial u_i}{\partial r_i})^T \Delta r_i \tag{4.5}$$

where $\frac{\partial u_i}{\partial r_i} = -1$ and $\frac{\partial u_i}{\partial \mathbf{v}_i} = -\frac{(\mathbf{x}-\mathbf{v}_i)}{(u_i+r_i)}$

$$\frac{\gamma E}{\lambda^2} = \frac{1}{4\alpha\sum_{i=1}^{n}\log(\frac{1}{y_i})(\frac{y_i}{1-y_i})^2(\frac{1}{u_i+r_i})^2[\|\mathbf{x}-\mathbf{v}_i\|^2 + (u_i+r_i)^2]} \tag{4.6}$$

Thus the learning rules become

$$\Delta \mathbf{v}_i = \frac{(\frac{y_i}{1-y_i})(\frac{u_i}{u_i+r_i})(\mathbf{x}-\mathbf{v}_i)}{2\alpha\sum_{i=1}^{n}log(\frac{1}{y_i})(\frac{y_i}{1-y_i})^2(\frac{1}{u_i+r_i})^2[\|\mathbf{x}-\mathbf{v}_i\|^2 + (u_i+r_i)^2]} \tag{4.7}$$

$$\Delta r_i = \frac{(\frac{y_i}{1-y_i})u_i}{2\alpha\sum_{i=1}^{n}\log(\frac{1}{y_i})(\frac{y_i}{1-y_i})^2(\frac{1}{u_i+r_i})^2[\|\mathbf{x}-\mathbf{v}_i\|^2 + (u_i+r_i)^2]} \tag{4.8}$$

Even for detection of ellipses and straight lines we can start with learning (v_1, v_2, r) till a limited error. However if number of straight lines in the image are known separately, we can start with constraining the matrix $\mathbf{A}$ as in eqn. (3.6) for those segments and for the rest of the segments learning the parameters of the circle. For elliptic model S_A we start with this center (v_1, v_2), assume cluster radius as r and initialise matrix $\mathbf{A}$ to $\frac{1}{r}\mathbf{I}$ where $\mathbf{I}$ is an identity matrix.

A partial ellipse segment may be identified as a segment of a hyperbola, which is a perceptible pattern. To avoid any extra segment from other the half, which is not a meaningful segment, we can apply another constraint on

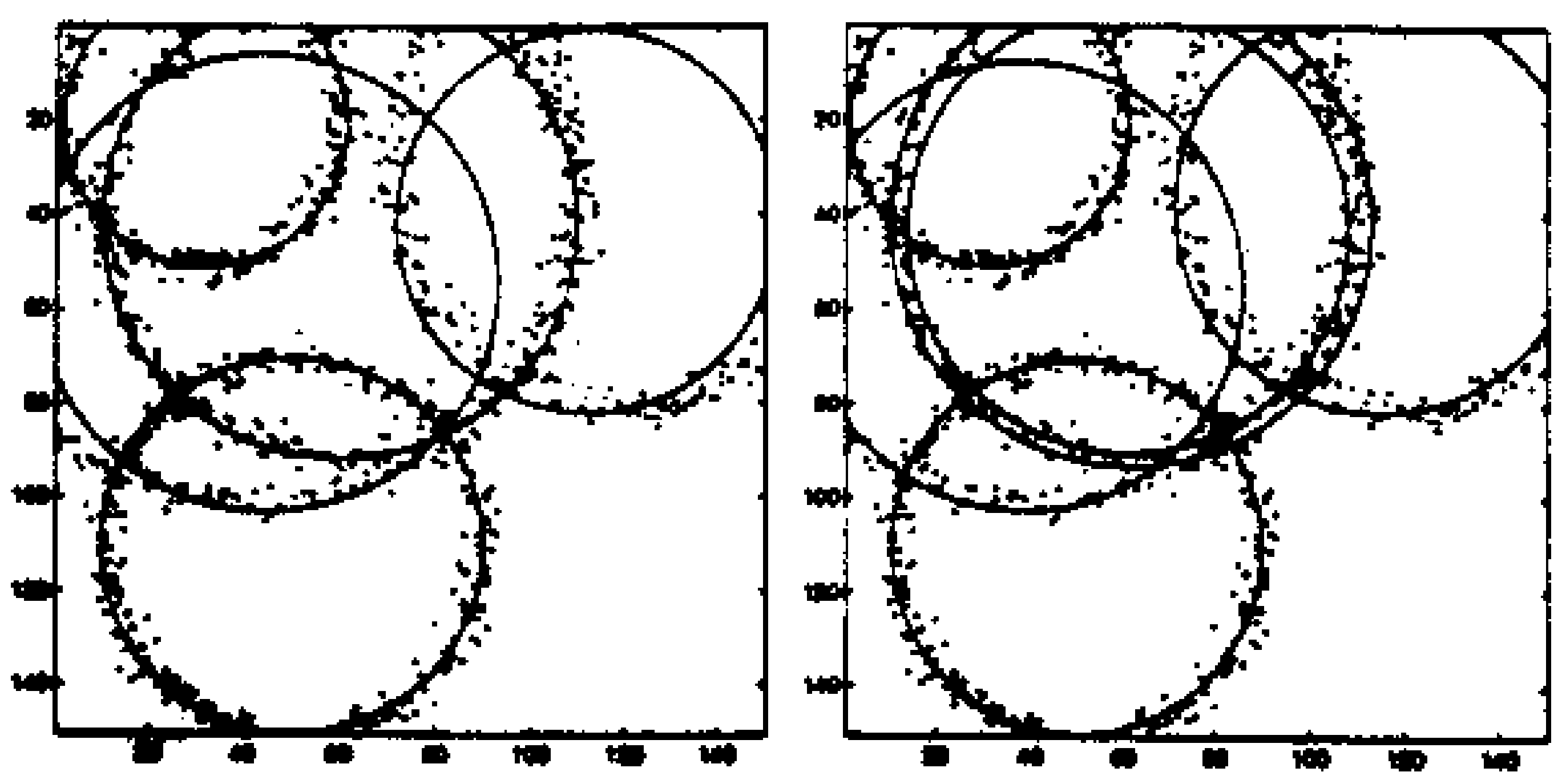

Figure 4.1: Output of network with excess no of output node

the matrix **A**. To force the matrix **A** to be positive definite, we introduce a matrix **B**, such that $\mathbf{BB}^T = \mathbf{A}$. In this case the learning rule becomes

$$\Delta\mathbf{B}_i = -\beta\frac{\partial E}{\partial \mathbf{B}} = -\beta\frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial u_i}\frac{\partial u_i}{\partial \mathbf{B}_i} \tag{4.9}$$

where $\frac{\partial u_i}{\partial \mathbf{B}_i} = \frac{1}{2(u_i+r_i)}[(\mathbf{x}-\mathbf{v}_i)(\mathbf{x}-\mathbf{v}_i)^T\mathbf{B}_i]$

The figure 4.1 shows a case where the no. of output nodes selected, was more than the number of segments expected in the image. Note that in most of the cases two of the segments detected by the network point to same segment.

According to level of difficulty we classify our problem into seven stages, in a hierarchical fashion.

1. Detection of circular shapes (Full & partial) from dense clustering.
2. Detection of a mixture of line and circle segments from dense clustering.
3. Same as 1,2 from sparse clustering.
4. Detection of complete & partial ellipses from dense clustering.
5. Detection of a mixture of ellipse and circle segments, or ellipse and line segments from dense clustering.
6. Detection of a mixture of lines and full or partial circular or elliptic shapes from dense clustering,
7. Same as 4,5,6 from sparse clustering.

The outputs of each classes are shown in figure 4.2-4.8 respectively.

During its convergence, the network may encounter several problems which are as follows.

1. Conflict between minor and major axis of a partial ellipse:- When only a partial information of an elliptic shape is present, there may be a conflict between the major and minor axis, where the converged ellipse chooses minor axis of original elliptic shape as its major axis or vice-versa. However both the results are perceptually acceptable. [fig 5.2:right]

2. Detection of spurious ellipses:- Spurious ellipses may be detected

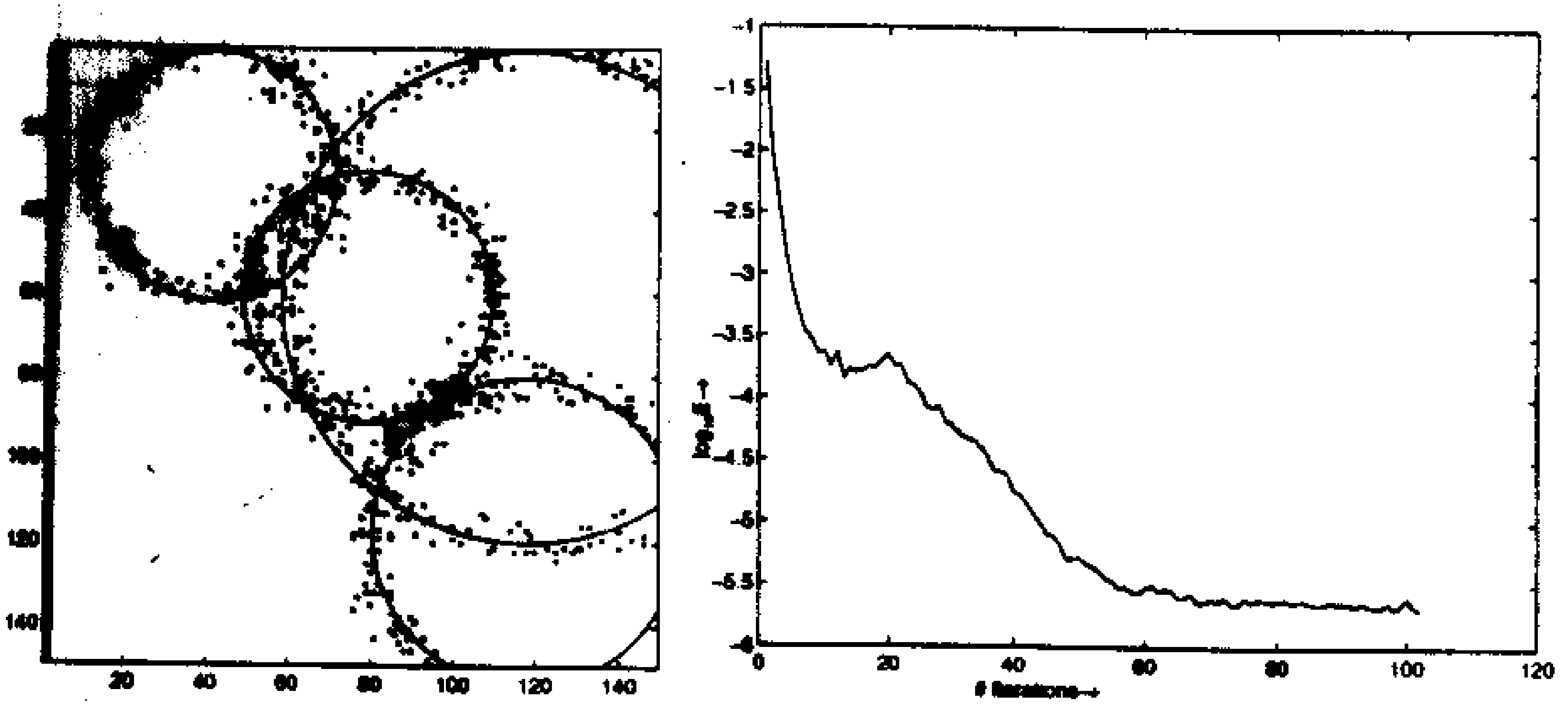

Figure 4.2: Case 1

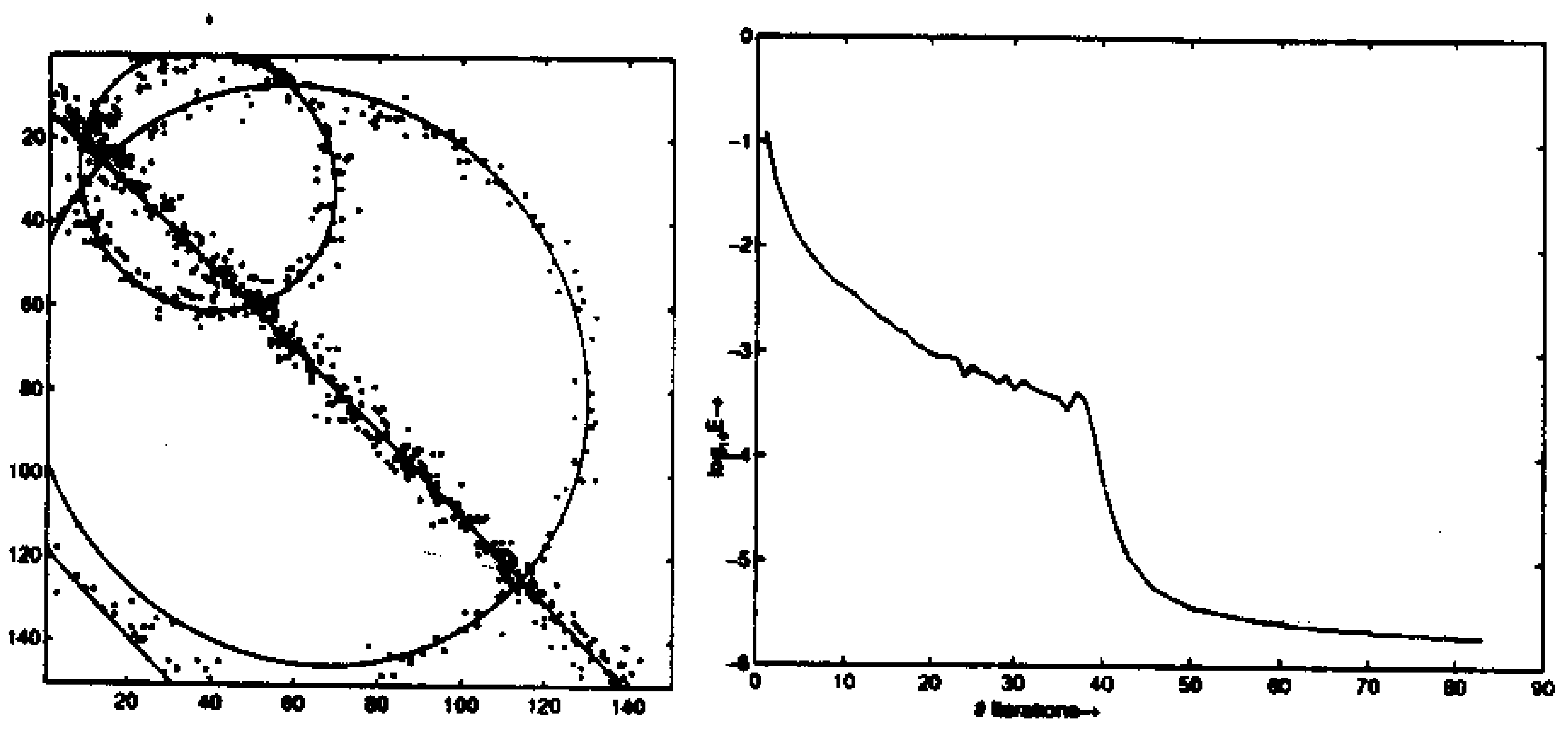

Figure 4.3: Case 2

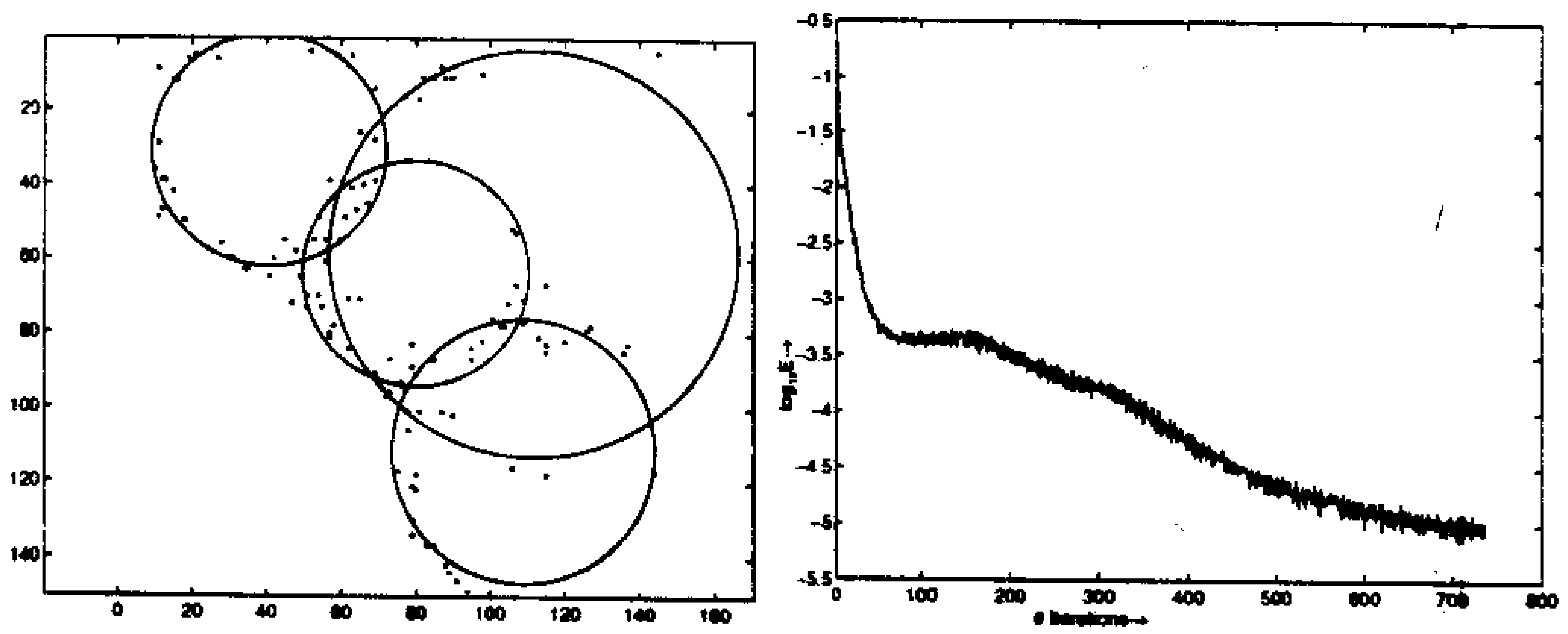

Figure 4.4: Case 3

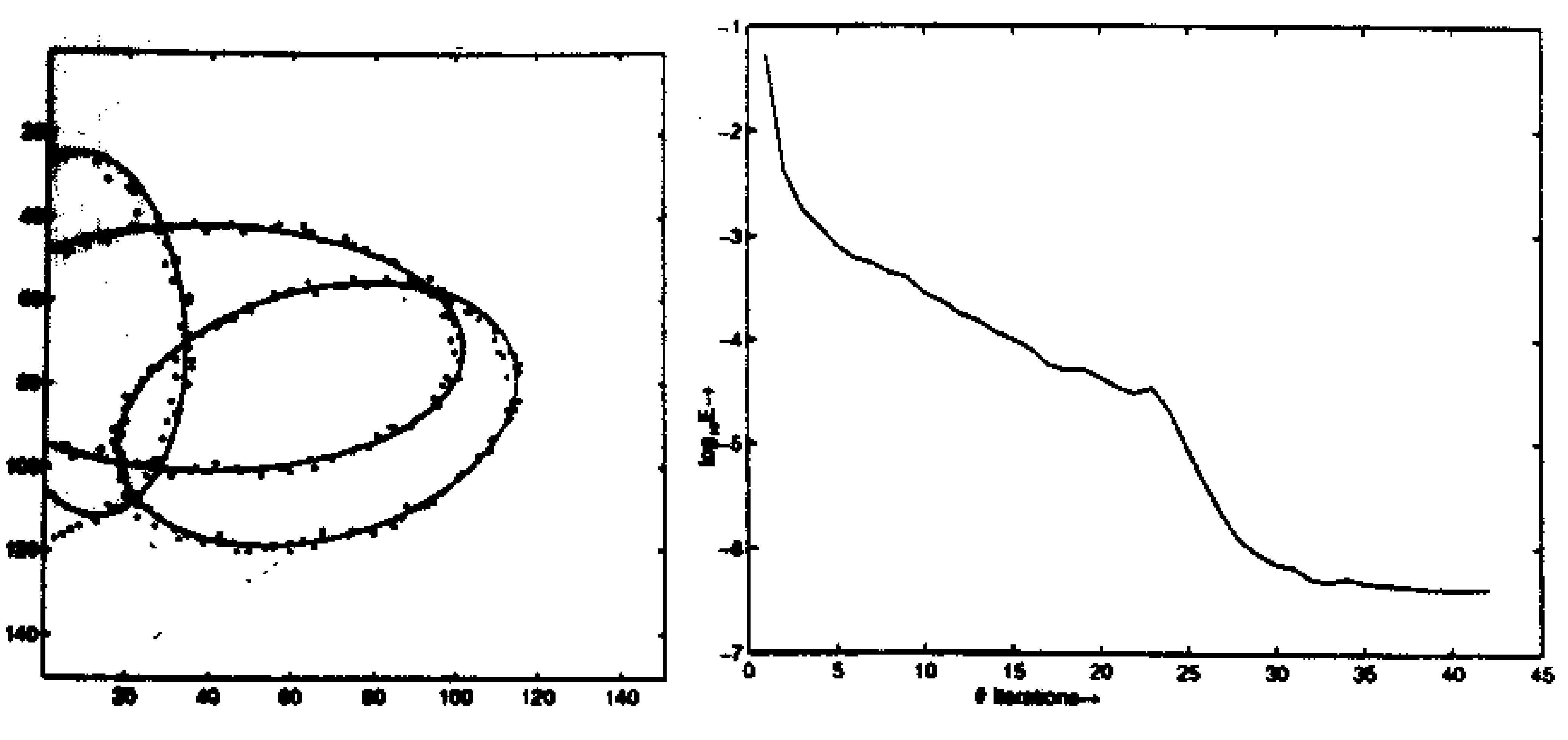

Figure 4.5: Case 4

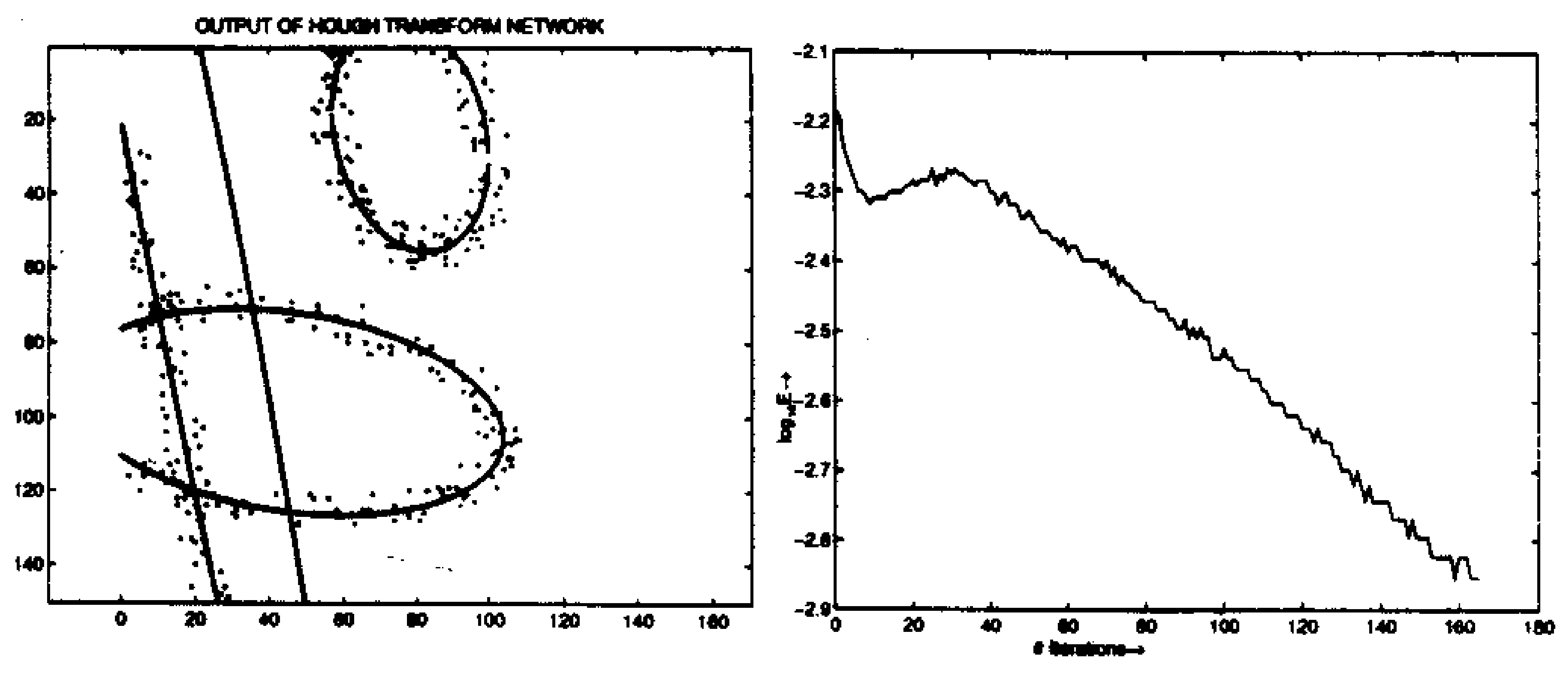

Figure 4.6: Case 5

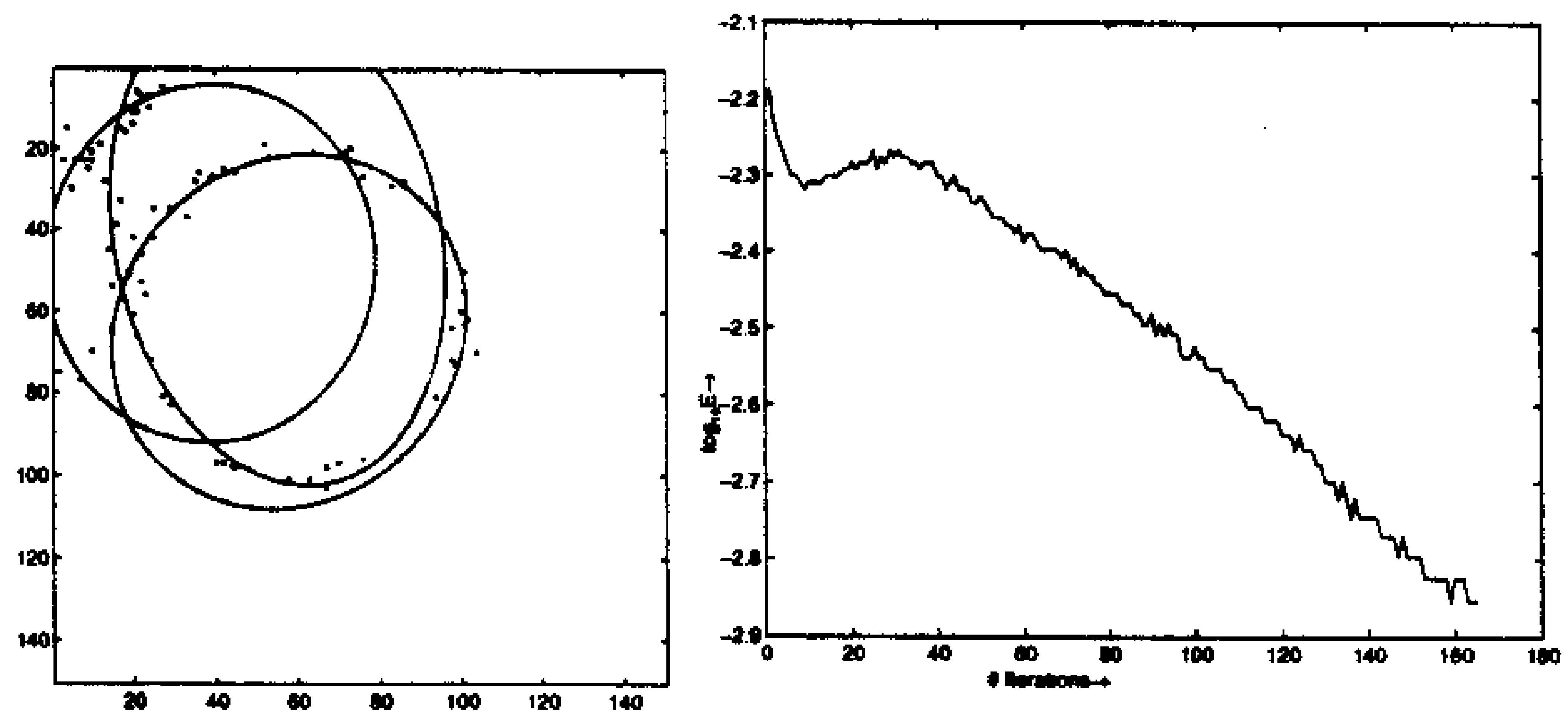

Figure 4.7: Case 6

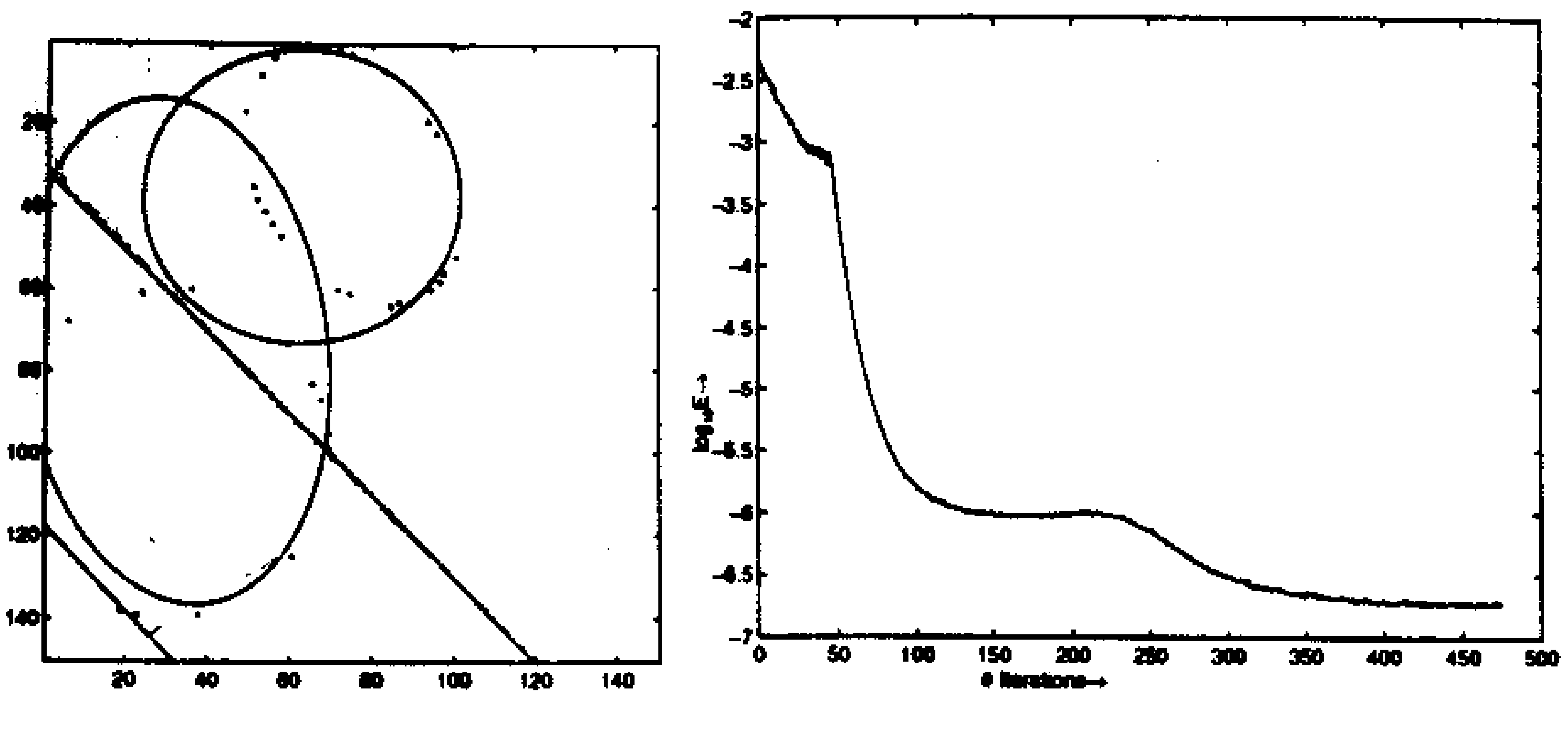
Figure 4.8: Case 7

intersection of several ellipses. However this may be a perceivable pattern.
3. Effect of Dominance: When a complete segment, having a large pixel count including cluster noise, is present side by side a smaller or partial segment, having a very small pixel count, the network may however sometimes ignore the smaller segment and moves to the larger one. In that case after complete convergence we have to check separately whether any more segment is present in the image and the network is run again ignoring the pixels nearing to the detected segments.

4.1.1 Local minimization error

It has been observed that the performance of the network is dependent on the situation of λ. For a large value of λ the detected parameter shapes interfere with each other and thereby causes poor localization. On the other hand, for a small value of λ initially it is very difficult to attract the distant points i.e. the basis of attraction of the solution space becomes very narrow. In order to take care of this situation initially λ is chosen a large value and then it is decreased as $\frac{\lambda}{\log(1+\tau)}$ where τ is the number of iteration at the stage. Note that, this is analogous to scheduling the temperature in simulated annealing and also the scheduling the mutation probability as in the case of genetic algorithms. Initial value of λ is taken as 45 and it is reduced up to 10. Few more outputs are in figure 4.9-4.11.

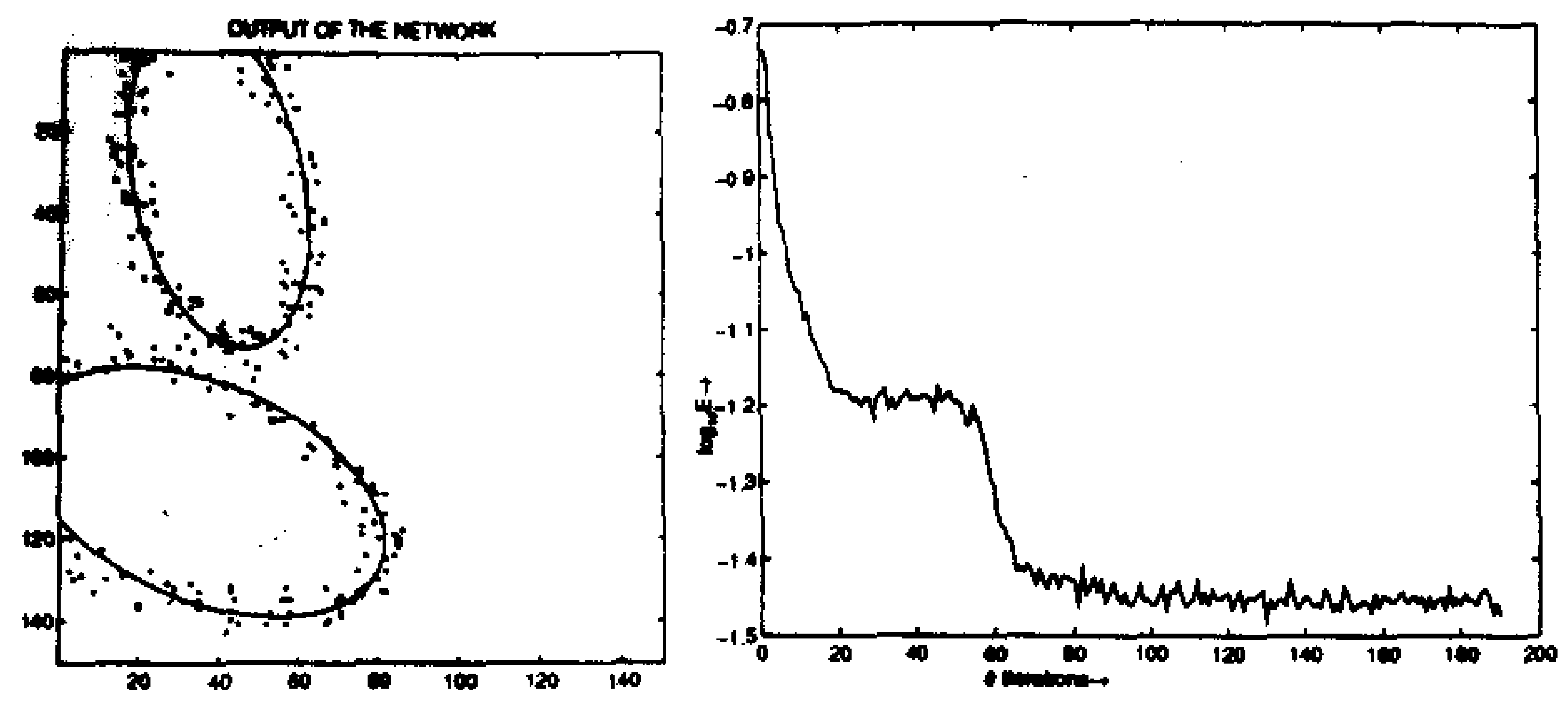

Figure 4.9: Typical example with error plot

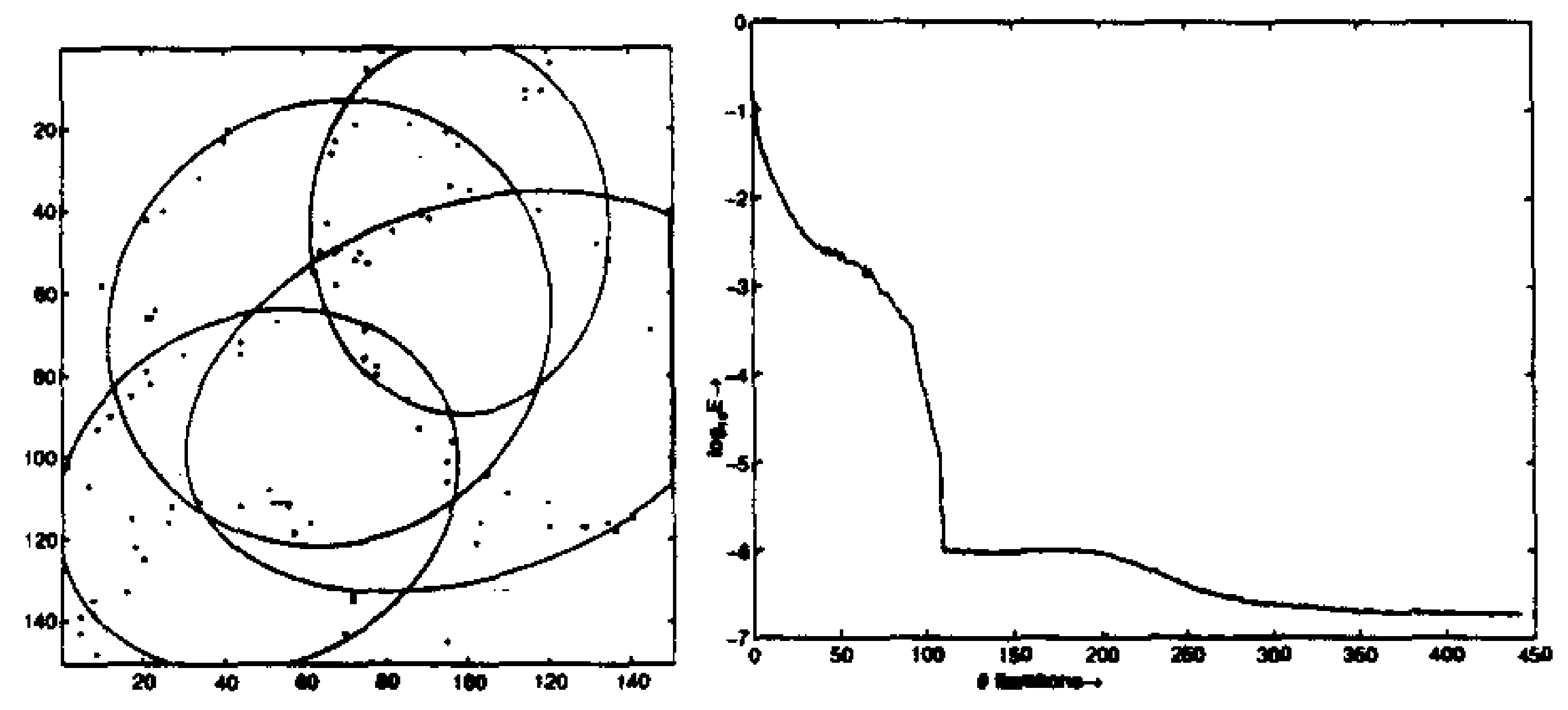

Figure 4.10: Typical example with error plot

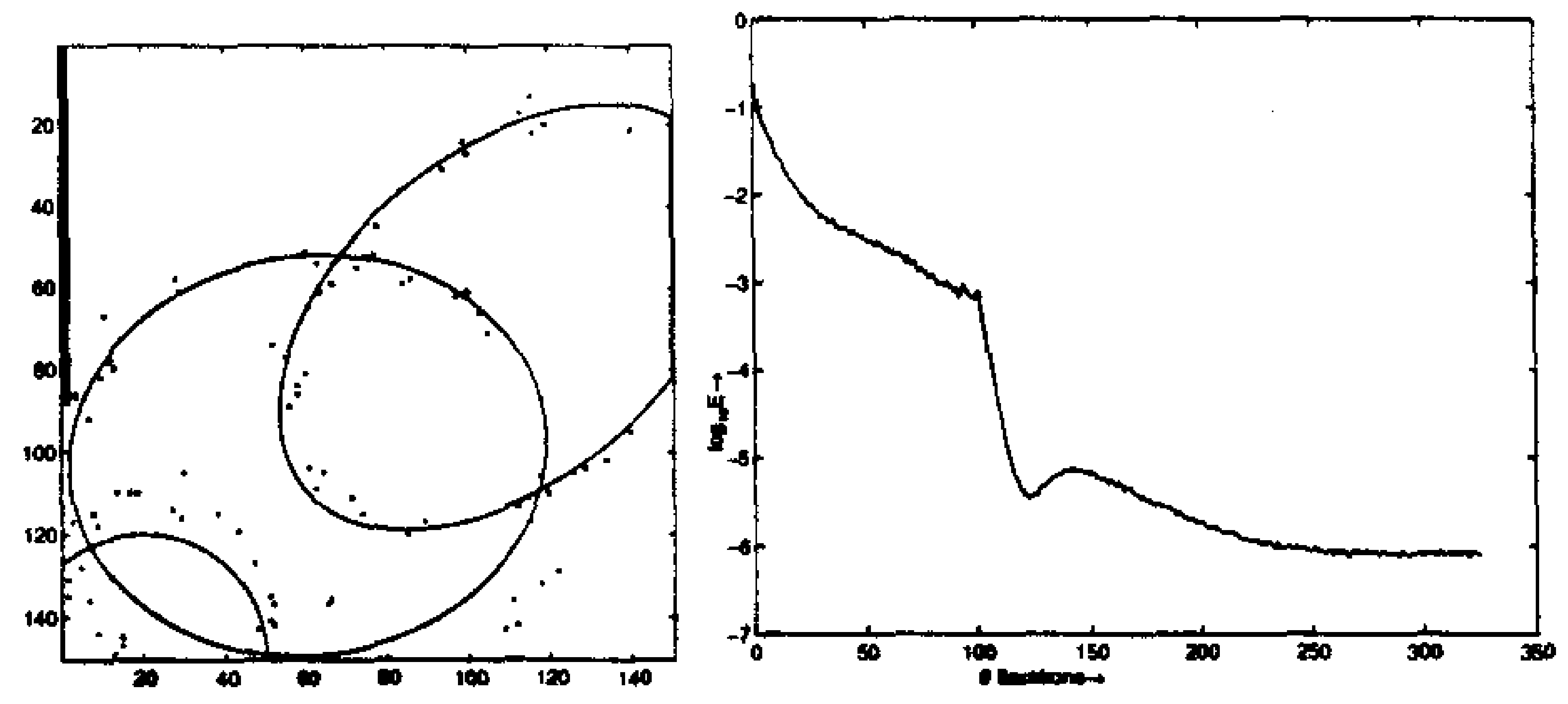

Figure 4.11: Typical example and error plot

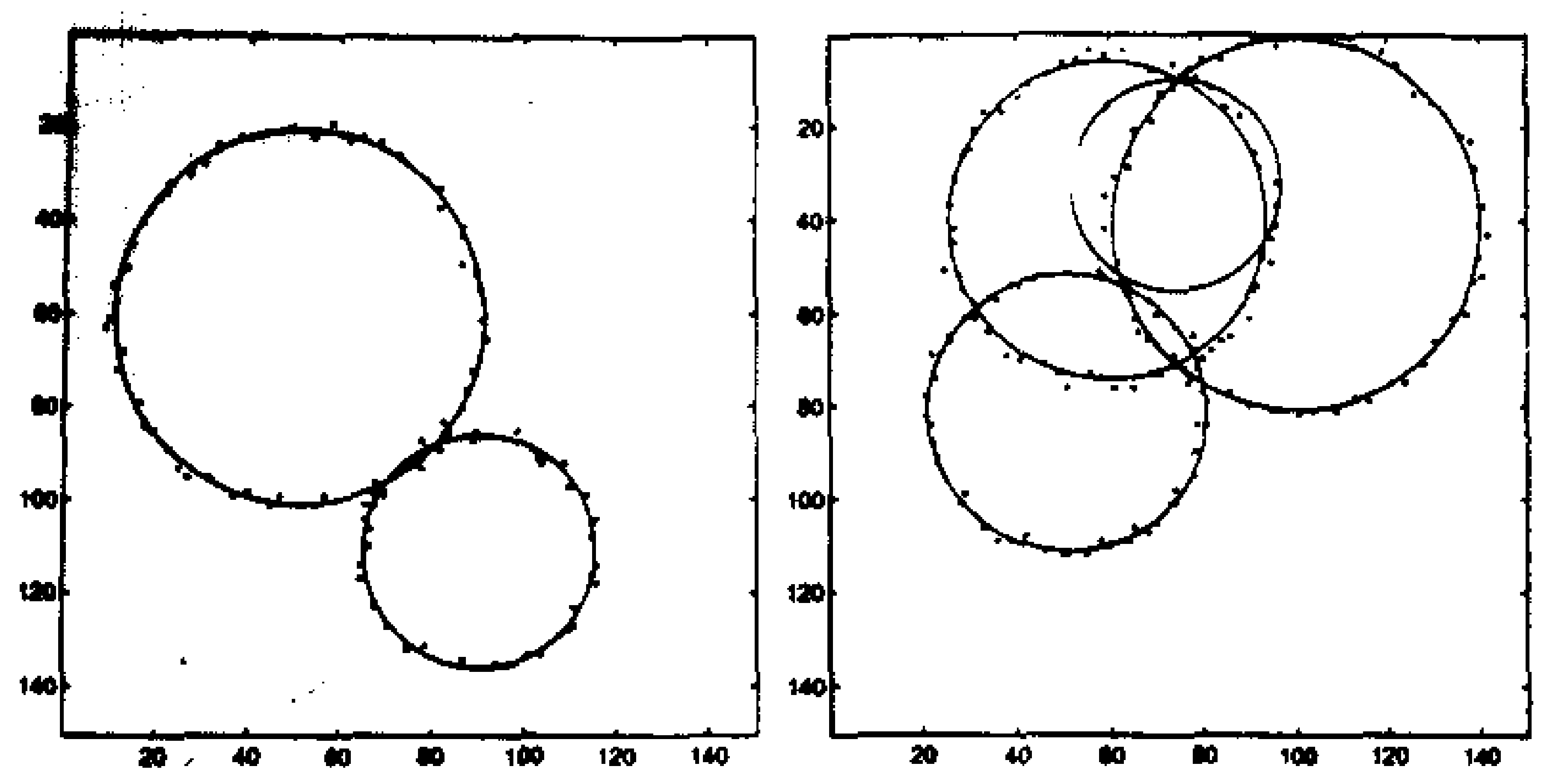

Figure 4.12: Applying classical Hough transform

4.2 Comparisons

4.2.1 Comparison with classical Hough transform

CHT is implemented for detection of circular shapes, using 3 parameters. In a noiseless image as in fig 4.12:left we see that CHT has detected the parameters correctly. However in case of fig 4.12:right it detects a spurious circle. In case of 10% noisy image as in fig 4.13 it has detected a few spurious circles, but none were correct.

CHT requires memory in the order of image space and hence it is unrealistic. For a typical 256×256 image, if we want to detect an elliptic shape, and if we assume accuracy of two pixels in center location, two pixels in radii determination and 2 degrees in angle calculation, and if no prior information about the range of these parameters are available, the accumulator size will be 128×128×64×64×45 i.e. 6000MB using short integer array. Thus space requirement increases exponentially with no. of parameters and the calculation becomes intractable.

4.2.2 Comparison with Randomised Hough transform

RHT [6,41] accumulates points in a parameter space by randomly choosing n-tuples of pixels from an image and computing the parameters of the object which passes through these pixels. To detect a circular shape, from the feature set we choose 3 points randomly and at each point local tangent is found out by the method of least squares. Parameters of the unique circle which pass through these 3 points is determined. If lines joining the center and those 3 points remain perpendicular to local tangents and the center lie on the line joining the midpoint of any two selected points and the point where tangents at those points meet, we accept those parameters, else we

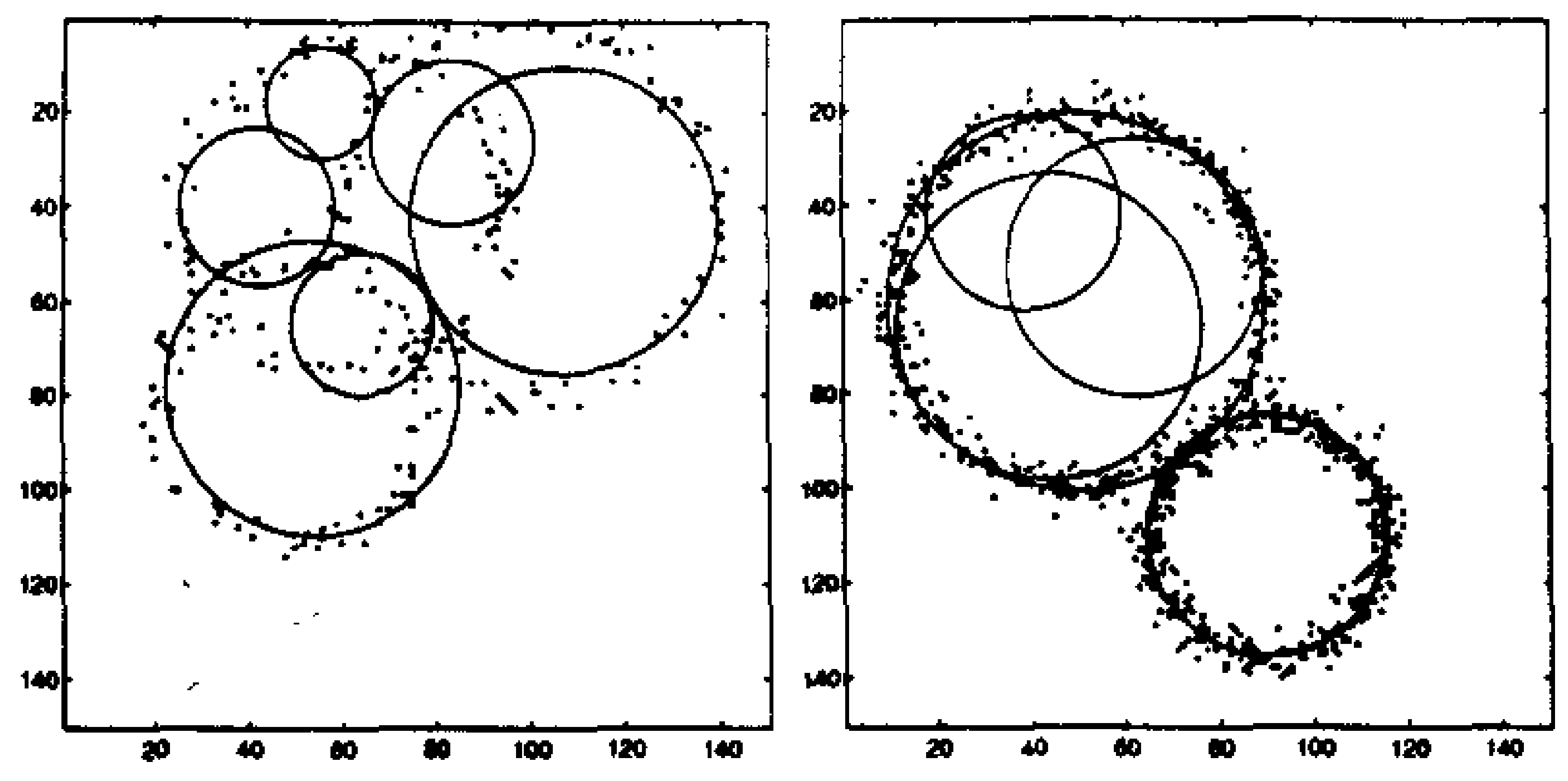

Figure 4.13: Detected by CHT: too many spurious outcome

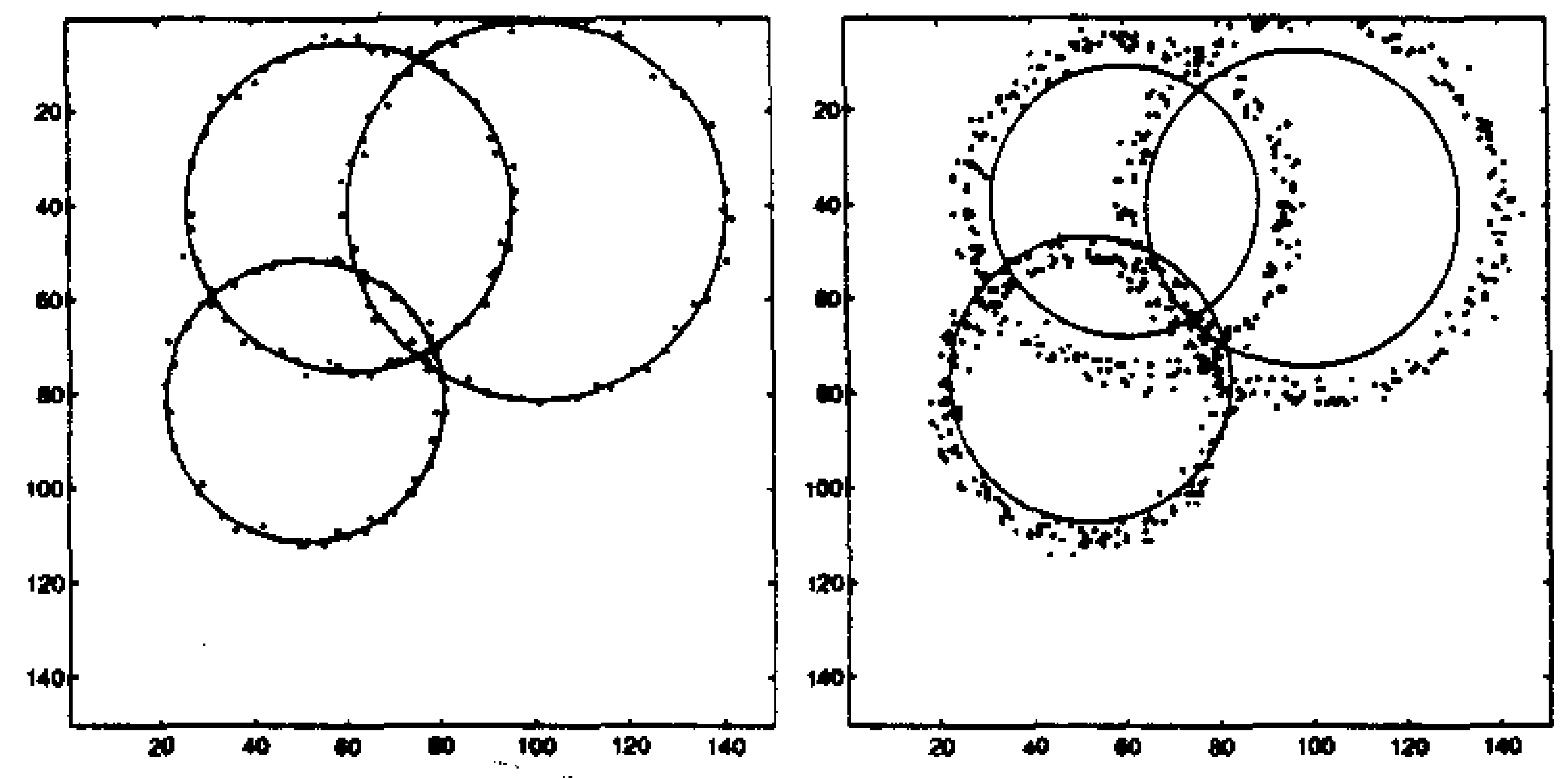

Figure 4.14: Detected by RHT: sparse clustering (precision may be low)

reject the 3-tuple and choose new one.

Fig 4.14:left shows the output of RHT in case of sparse clustering. Since the output depends only on final chosen tuples, the precision may not be high as in figure 4.14:right. Also this algorithm may work poorly when cluster noise is added to the shapes as in figure 4.15, because it makes increasingly difficult to find local tangents, the problem is severe at crosspoint of segments.

4.2.3 Comparison with Adaptive fuzzy c-shell clustering algorithm

$\star T$=elapsed cpu time.
$\star P = \frac{1}{N}(\prod_{i=1}^{N}(1 - y_i))$,where N=no. of object pixels. $\star\lambda = 45$ is the basis.

Based on the definition(1) of the prototype and the distances a c-shell functional is formulated [2]. Given a set of n points, $\mathbf{x} \equiv \{x_1, x_2, ..., x_n\} \subset \Re^p$. where x_k is the k^{th} feature vector in set X the problem of detection of ellipse segments boils down to finding c-cluster prototypes as well

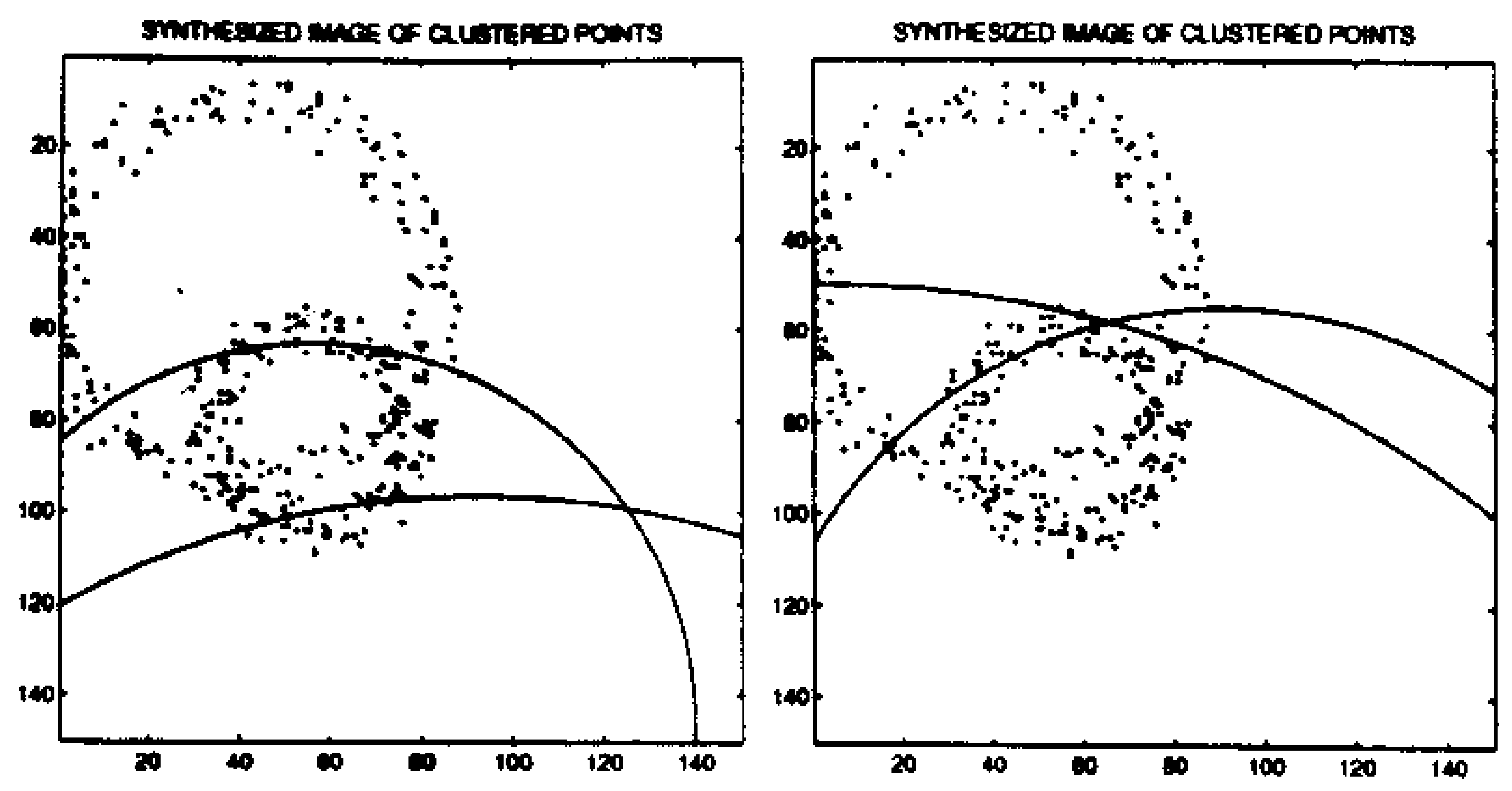

Figure 4.15: Detected by RHT: dense cluster noise leads to failure

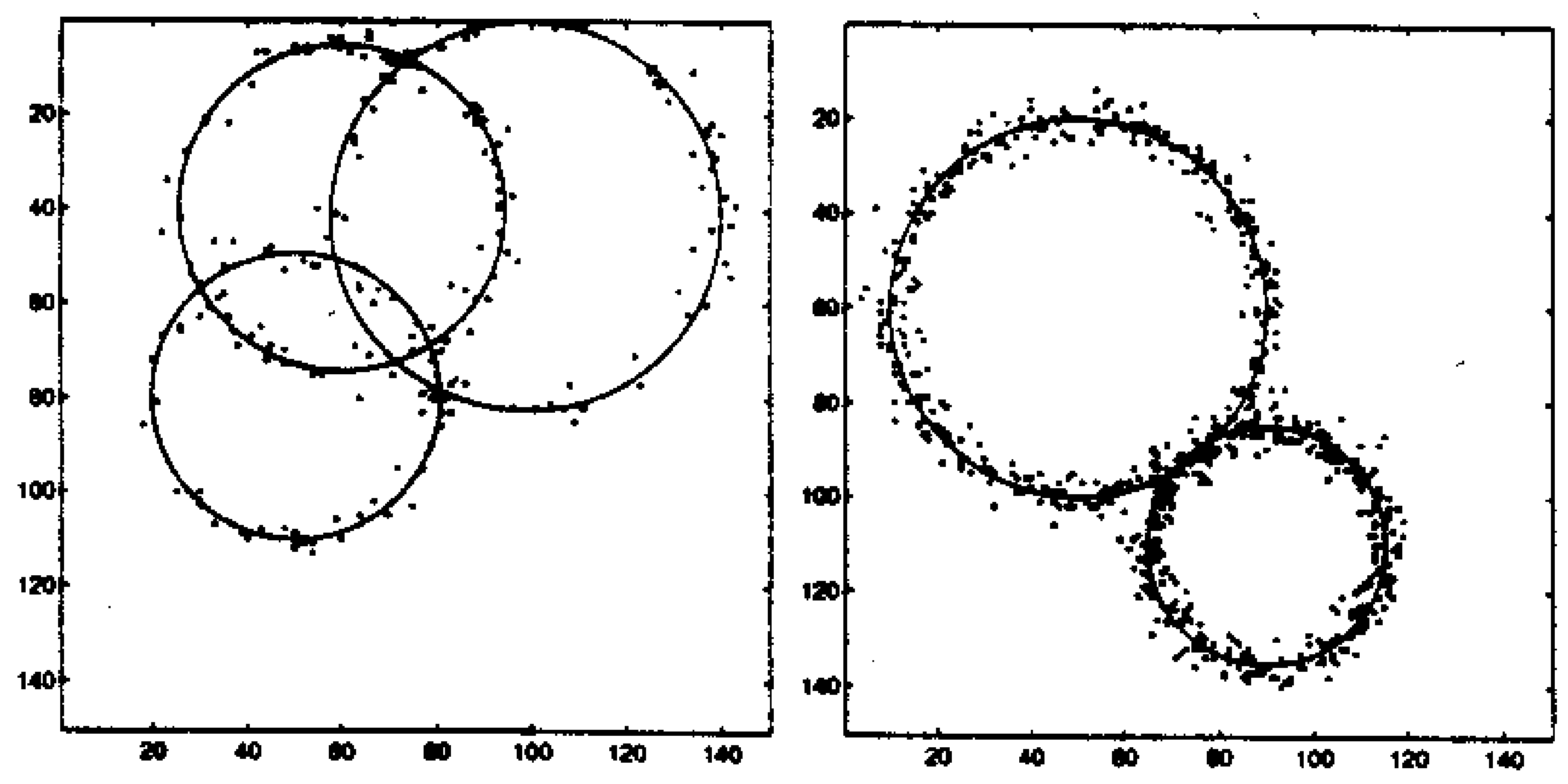

Figure 4.16: Detected by Network: same images

Cases	Noise in Maximum shift of pixels									Comments
			±0	±2	±4	±6	±8	±10	±12	
Case1	#pixels		193	461	533	598	740	811	866	only
	Network	T	57.55	68.02	81.95	98.76	102.66	68.76	71.90	circles
		P	4.162e-5	6.079e-5	7.119e-5	8.059e-5	4.72e-4	6.818e-4	7.606e-4	(dense)
	AFCS-U	T	68.33	84.40	109.97	246.66	375.58	410.11	511.19	
		P	4.205e-5	1.004e-4	2.145e-4	5.653e-4	2.119e-3	3.244e-3	5.364e-4	
Case2	#pixels		242	334	456	480	537	603	664	circles
	Network	T	74.17	98.33	146.93	199.51	190.45	210.67	232.29	+ line
		P	1.713e-4	2.506e-4	4.110e-4	6.002e-4	7.023e-4	7.910e-4	8.004-4	(dense)
	AFCS-U	T	68.53	96.41	183.22	211.37	288.01	321.29	378.92	
		P	1.864e-4	3.499e-4	5.606e-4	7.070e-4	8.922e-4	1.463e-3	2.243e-3	
Case3	#pixels		68	71	78	83	88	92	97	sparse
	Network	T	64.58	66.13	67.45	69.05	70.03	71.24	72.12	
		P	1.044e-5	2.674e-5	4.887e-5	6.099e-5	8.225e-5	1.384-4	1.969e-4	2.396e-4
	AFCS-U	T	84.22	95.33	101.12	124.56	148.88	190.31	245.22	
		P	1.083e-5	3.434e-4	5.609e-4	8.802e-4	1.566e-4	1.985e-4	2.466e-4	
Case4	#pixels		132	230	343	405	514	656	784	only
	Network	T	85.33	86.54	91.97	100.95	148.43	190.36	226.73	ellipse
		P	1.761e-4	3.142e-4	7.809e-4	1.104e-3	1.180e-3	1.222e-3	1.29e-3	(dense)
	AFCS-U	T	79.46	88.33	105.35	140.73	299.65	330.41	371.13	
		P	1.803e-4	3.543e-4	4.214e-4	5.455e-3	6.415e-3	8.223e-3	1.042e-2	
Case5	#pixels		311	357	410	447	523	607	684	ellipse
	Network	T	94.35	114.44	154.65	245.32	312.13	298.57	354.11	+ line
		P	3.421e-4	5.644e-4	7.403e-4	9.006e-4	1.525e-3	2.346e-3	4.537e-3	(dense)
	AFCS-U	T	84.11	119.38	178.35	267.44	342.39	412.79	489.56	
		P	4.112e-4	6.344e-4	9.190e-4	1.225e-3	3.906e-3	5.977e-3	7.107e-3	
Case6	#pixels		142	167	178	230	263	293	324	any
	Network	T	84.74	88.31	102.55	142.74	190.38	261.11	319.54	(dense)
		P	3.865e-4	5.453e-4	8.549e-4	1.002e-3	1.232e-3	1.476e-3	1.932e-3	
	AFCS-U	T	78.33	103.42	178.56	310.32	446.62	567.73	611.11	
		P	4.210e-4	5.439e-4	9.769e-4	2.233e-3	4.535e-3	7.326e-3	8.435e-3	
Case7	#pixels		42	54	67	80	93	103	124	any
	Network	T	84.74	88.31	102.55	142.74	190.38	221.11	239.54	(dense)
		P	1.223e-4	1.323e-4	1.466e-4	1.575e-4	1.656e-4	1.895e-4	2.021e-4	
	AFCS-U	T	63.12	70.33	203.12	288.12	345.24	411.22	575.35	
		P	4.355e-5	4.644e-5	1.253e-4	1.674e-4	1.943e-4	2.311e-4	4.211e-4	

as a c-partition of X. The prototypes are defined using v_i from the set $V \equiv \{v_1, v_2, ..., v_c | v_i \in \Re^p\}$, and r from the set $R \equiv \{r_1, r_2, ..., r_c | r_i \in \Re^+\}$. The partition may be either hard or fuzzy. Let M_{fc} and M_c denote sets of all hard and fuzzy c-partitions respectively with $U \in M_f$ or $U \in M_c$. Then the functional J_s is defined as

$$J_s(U, V, R, A) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m (D_{ik})^2 \qquad (4.10)$$

where u_{ik} is the membership of the k^{th} feature vector in the i^{th} cluster, with exponent $m \in [1, \infty]$. If the membership is fuzzy then $u_{ik} \in [0, 1]$;otherwise $u_{ik} \in \{0, 1\}$. Assuming that A_i is fixed, the sets U, V, R are the variables of the functional, and the functional is minimised with respect to them. A constraint is imposed on the membership as follows:$\sum_{i=1}^{c} u_{ik} = 1$ Based on the definition (2) and (3) of prototype and distances the functional of (4.10) is written as

$$J_s(U, V, A) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m (D_{ik})^2 \qquad (4.11)$$

Thus the functional is optimized with respect to u_{ik}, $\mathbf{v}_i$ and $\mathbf{A}_i$. Following the optimization conditions, for cluster centers $\mathbf{v}_i$ and norm-inducing matrices $\mathbf{A}_i$ for either case of memberships

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (x_k - v_i) = 0 \qquad (4.12)$$

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (x_k - v_i)(x_k - v_i)^T = 0 \qquad (4.13)$$

are satisfied.

The comparison shows that, AFCS-U algorithm works efficiently, when number of object pixels are less and noise level is low. It can recognise partial segments with exceptional efficiency. However in presence of noise the algorithm performs poorly, and when the no. of object pixels are very high as well as noise is present, performance and precesion is low. The performance is also poor in complicated cases of overlapped ellipse segments where fuzzy-c-means and linear AFCS fail to provide good initialization to AFCS-1 algorithm.

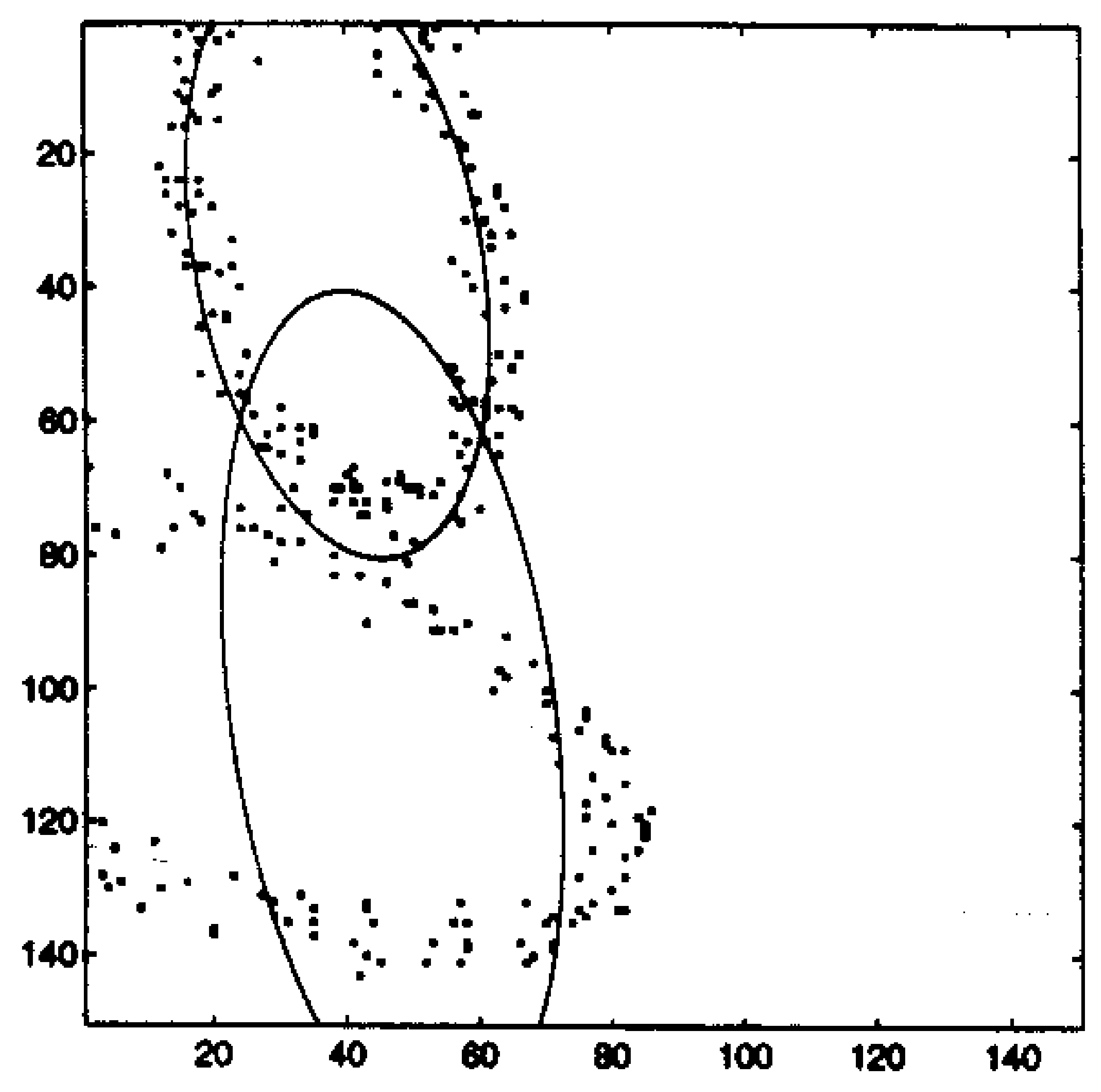

Figure 4.17: Failure of AFCS-1 to provide good initialisation to AFCS-U : high sensitivity to noise

Chapter 5

Extension of the Network to Gray Level

The connectionist model is extended in order to detect conoidal shapes in gray level images. Conventionally the darkest pixels have the lowest gray value, and vice versa. The normalized gray values of the pixels are scaled in the range [0-1] and divided into 3 partitions on the basis of preselected thresholds. Note that, the thresholds selected may vary for different images, particularly depending on the quality and contrast. For example, if the chosen thresholds are th_1 and th_2 then the gray values in the three partitions are $[0, th_1); [th_1, th_2); [th_2, 1]$.

The network structure for this pattern is modified as figure 5.1, where number of nodes in the input layer becomes 3 and all the input nodes are connected to output nodes. Thus the input layer takes 3 parameters from the image (viz x_1, x_2, g), where g is the normalised gray level pixel value. The output of the network is calculated as $y_i = g_i f(u_i)$ instead of eqn. (2.3).

The pixels belonging to the first partition (containing the darkest pixels of the image) are fed to the network first keeping the learning rules same as (3.22) & (3.23), since constant normalised gray value g_i is assumed. After the network converges with a sufficient number of input patterns, we go for the second partition. The normalised gray values in the second partition $[th_1, th_2)$ are again subdivided into a set of smaller partition $[th_1, th_1 + \epsilon); [th_1, th_1 + 2\epsilon); ...[th_2 - \epsilon, th_2)$. The object pixels in the sub-partition $[th_1, th_1 + \epsilon)$ are then appended to those in the first partition and the network is retrained. After convergence the pixels in the second sub-partition ($[th_1+\epsilon, th_1+2\epsilon)$) are also fed to the network along with the pixels in $[0, th_1+\epsilon)$. The process is thus continued by feeding pixels in sub-partition in regular intervals along with current pixels. This method is adopted in order to reduce the computation time. The object pixels belonging to the third partition is simply ignored. Thus, darkest object pixels are used maximum number of times for training purpose. The experimental results are shown in figure [5.2-5.3].

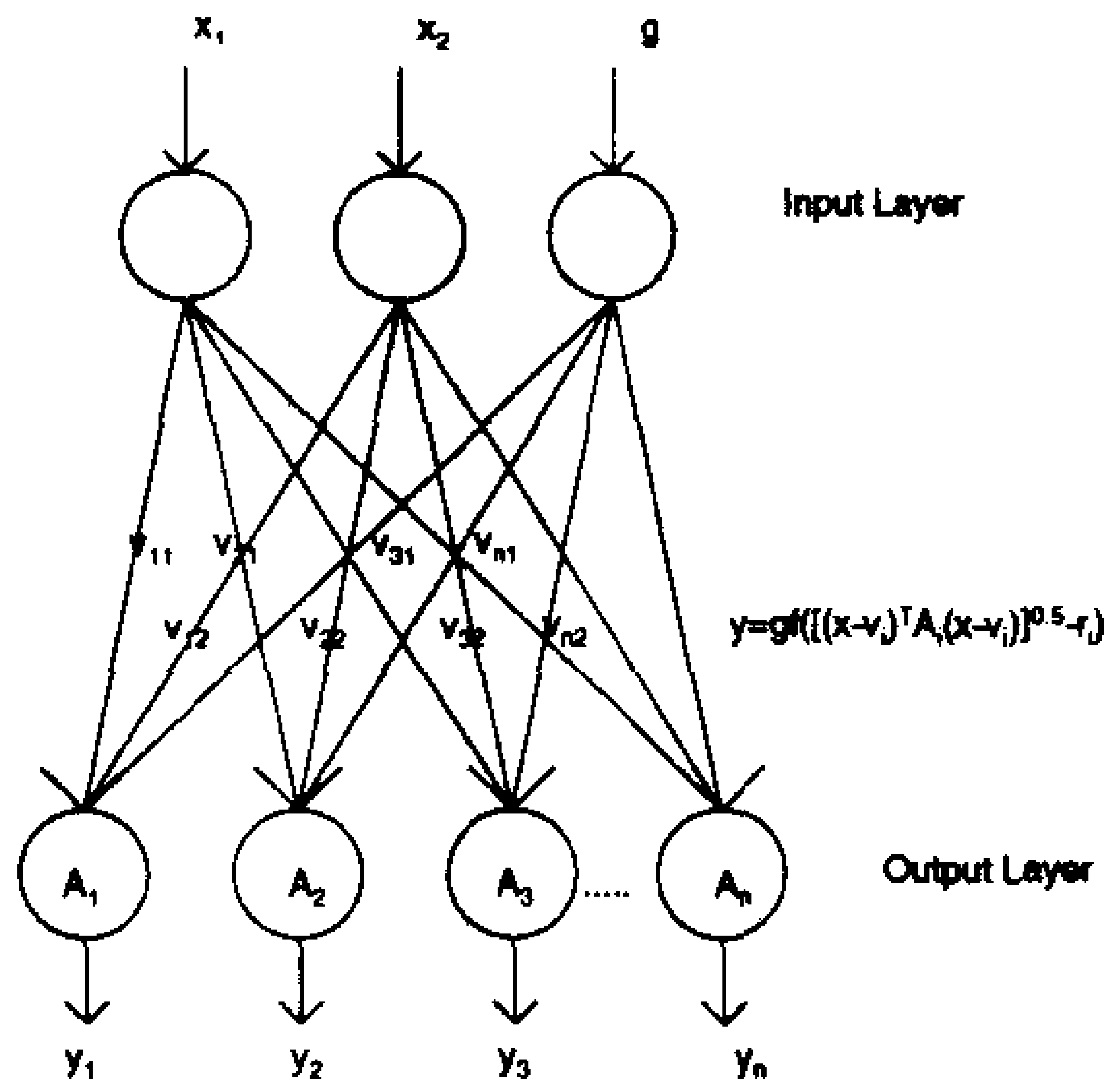

Figure 5.1: Network Architecture for detecting shapes in gray level images

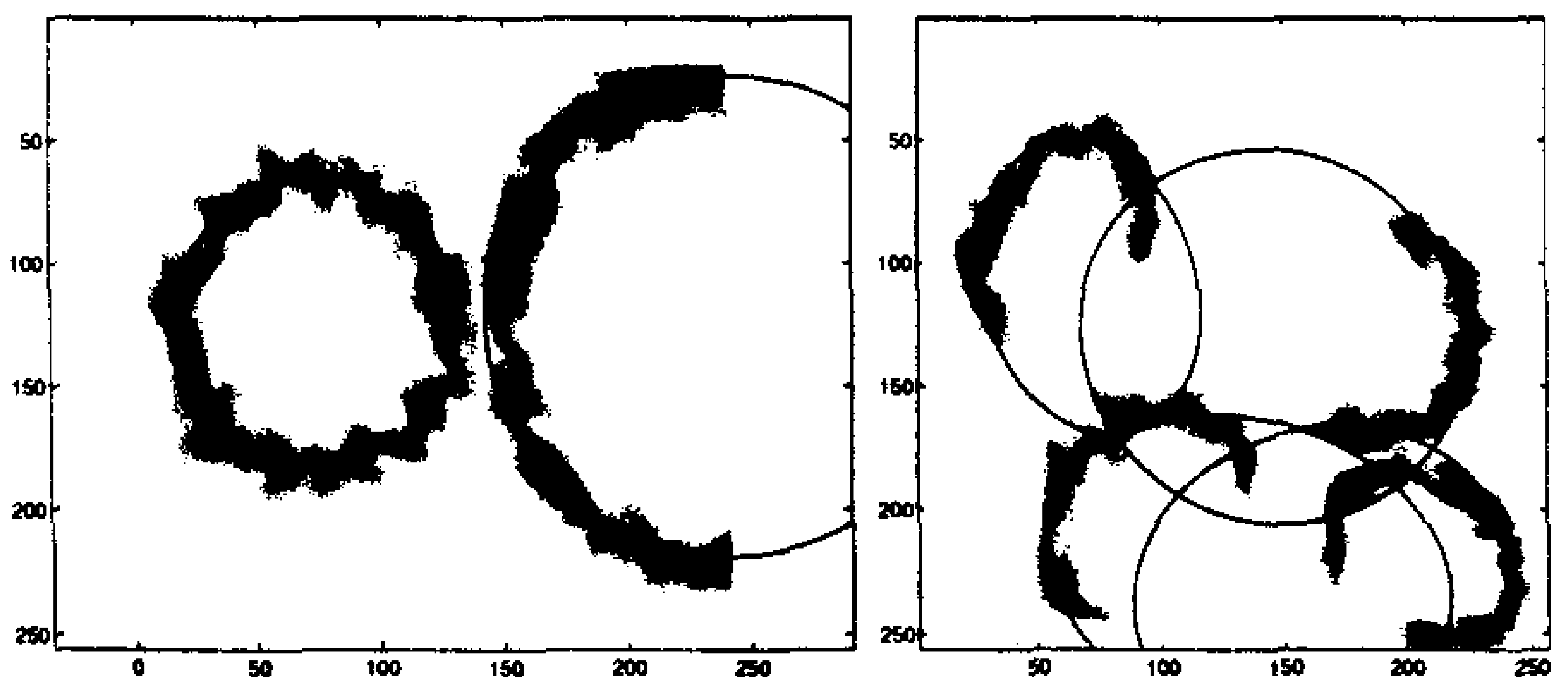

Figure 5.2: Applying to gray level images

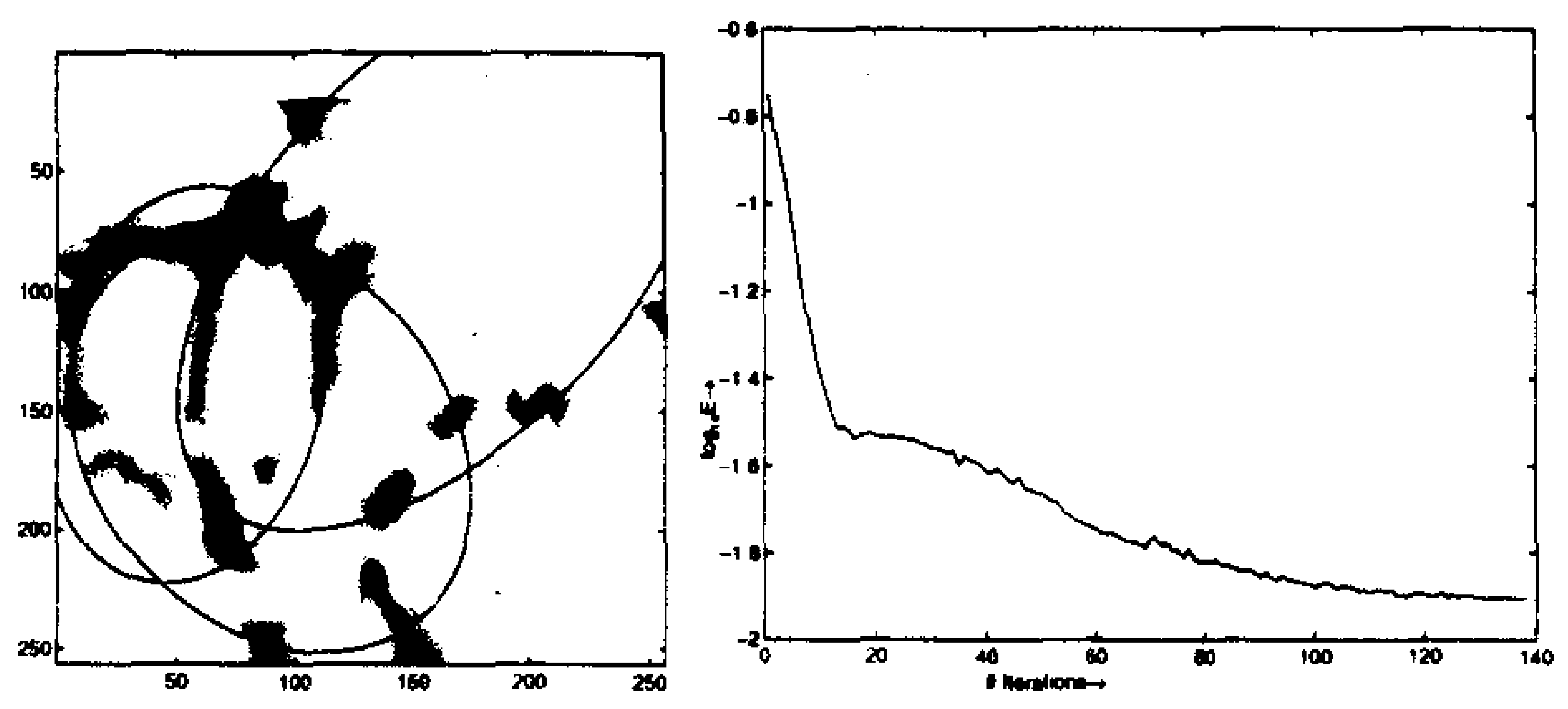

Figure 5.3: Applying to gray level images

Chapter 6

Discussion and Conclusions

The network model designed here is able to learn the parametric forms of various parametric shapes in unsupervised manner and to provide an efficient model for learning and representation of visual information. Its performance is compared with those of some existing algorithms like classical HT, randomised HT, AFCS-U algorithm and it is found that it excels over these algorithms in many cases. Particularly in noisy environment the present algorithm works gracefully than the other algorithms. The network is able to detect any conoidal structure in a gray image, and it can also be used to detect partial segments from a very sparse clustering. The parameters are represented here with a very high precision, because unlike classical Hough transform this method does not need any quantization of the parameter space.

The network however has its own limitations. A few examples are shown, (noise level is ± 15 pixels), where the network falls behind ideal human vision system 6.1. To improve capability of the network, certain co-operative computational mechanism may be incorporated.

Besides this, other future scopes of investigations include, (*i*) providing the justification of the learning ratios, (*ii*) extending the network in multi-dimensional models on gray level, (*iii*) demonstrating its other applications like detecting sinusoidal shapes, boundaries and reflectional symmetry.

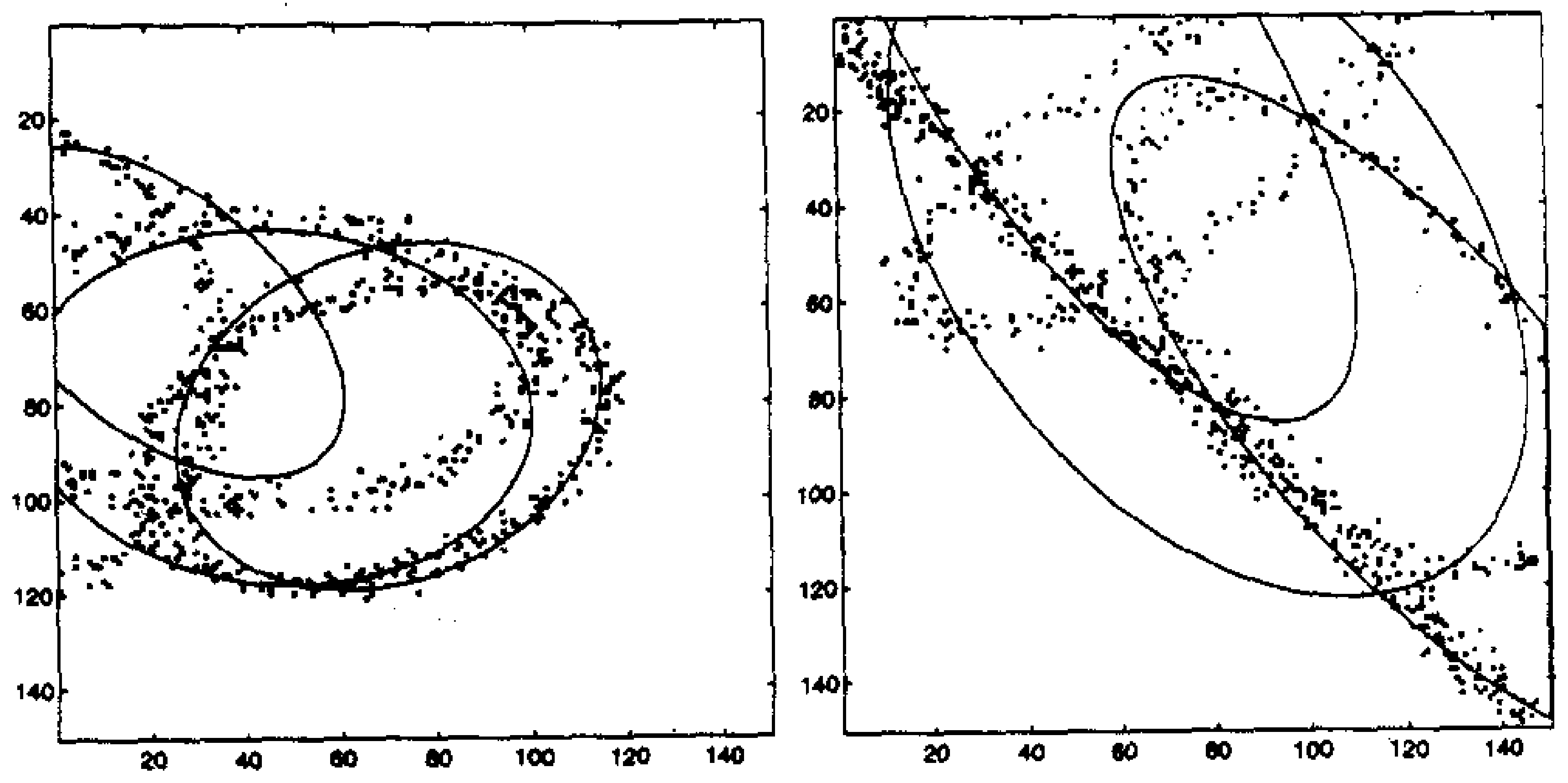

Figure 6.1: Failures of the network

Appendix A

Algorithms

```
Algorithm A.0.1: CHT(x,y)

comment: Computes classical Hough transform
global
$N$ = No. of object pixels in the image
$K_{threshold}$ = Threshold for accumulator array
$output = \varnothing$
$RESOLUTION_k$ = Resolution of $\theta_k$ in parameter space
$A(\theta) \leftarrow 0 \forall \theta \in$ parameter space
$(x_i, y_i) = i^{th}$ image pixel coordinates
for each $i \in N$
  do
    compute $\Theta$ such that $f(x_i, y_i, \Theta) = 0$
    for $k \leftarrow 1$ to $size(\theta)$
      do $\theta_k \leftarrow round(\Theta_k / RESOLUTION_k)$
    $A(\Theta) \leftarrow A(\Theta) + 1$
for each $\theta \in$ parameter space
  do
    if $A(\Theta) > K_{threshold}$
      then output $\leftarrow$ output $\cup \Theta$
```

Algorithm A.0.2: YUEN HT(X)

comment: Detects Ellpses in image using Yuen's modification of HT

The Hough transform algorithm stated below, does not consider the case of concentric ellipses at different orientations, but avoids too many computations. The technique mentioned here utilizes the center finding approach, along with a version of the backprojection technique for peak sharpening and feature point labelling.

For finding the rest of the parameters Muammar and Nixon's tristage Hough transform technique for multiple ellipse extraction is utilised. For finding orientation, a pair of points equidistant from the center but not having the same magnitude of the slope are considered. For concentric ellipses, having different orientations, there would be multiple dominant peaks in this histogram. Once centers and the orientation are found, a two dimensional accumulator array can be utilized to find major and minor radii. For concentric ellipses of different sizes, there would be multiple peaks in the array.
1.Accumulation: For each pair of points, find the line MT, and find the votes along the line in the accumulator array ACC1.
2.Peak Sharpening: For each point i, by inspecting all the cells that received contribution from all the pairs, that i was included in, find the cell location that received the maximum number of votes. Increment that location in a second accumulator array ACC2. Also note this location for point i in tracking array.
3.Peak detection: Search ACC2 for the highest peak. Repeat to find the next peaks by zeroing out the peak location along with its neighbourhood. until no peaks are present that meet a minimum threshold.
4.Feature labelling: For each peak, identify the points that contributed to the peak by examining the tracking array. This will segment the data set.
5.Orientation: Increment the orientation accumulation array for each pair of points equidistant from the centre. Find the highest peak, which gives the orientation of a candidate ellipse.
6.Major/minor radii: Increment the two dimensional radius array for each pair of points. Find all the dominant peaks.

Algorithm A.0.3: RunNetwork($\mathbf{X}, num_{segment}$)

```
comment: Computes Parametric shapes using 2-layer ANN
$count \leftarrow$ No. of object pixels in $\mathbf{X}$
$\lambda \leftarrow 45; \tau \leftarrow 1; count_c \leftarrow 1; \kappa \leftarrow 0.01; \eta_{err} \leftarrow 1; test_f \leftarrow 1; c \leftarrow 0.5$
for $i \leftarrow 1$ to $num_{segment}$
  do $\Delta A_i \leftarrow \mathbf{I}$
while $(\eta_{err})$
  do
    $\eta_{err} \leftarrow ConvergenceCondition(test_f, \tau, \eta_{err}, \lambda)$
    $\mathbf{X} \leftarrow RandomPermute(\mathbf{X})$
    $\tau \leftarrow \tau + 1$
    for $var \leftarrow 1$ to $count$
      do
        $\mathbf{x} \leftarrow \mathbf{X}(var, :)$
        for $i \leftarrow 1$ to $num_{segment}$
          do
            $u_i \leftarrow (\mathbf{x} - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x} - \mathbf{v}_i) - r_i$
            if $(imag(u_i) \neq 0)$
              do $u_i \leftarrow (\mathbf{x} - \mathbf{v}_i)^T (\mathbf{x} - \mathbf{v}_i) - r_i$
            $y_i \leftarrow exp(-(u_i/\lambda)^2)$/*Activation funtion*/
            $E \leftarrow [\prod(1 - y_i)]^\alpha$
            for $i \leftarrow 1$ to $num_{segment}$
              do
                $\mathbf{R} \leftarrow (\mathbf{x} - \mathbf{v}_i)$
                $\mathbf{M} \leftarrow -2\mathbf{A}_i\mathbf{R} + (\mathbf{A}_i - \mathbf{A}_i^T)\mathbf{R}$
                $\mathbf{W} \leftarrow 2\mathbf{R}\mathbf{R}^T - diag(\mathbf{R}\mathbf{R}^T)$
                $G_i \leftarrow (\mathbf{M}\mathbf{M}^T) + trace(\mathbf{W}^T\mathbf{W})$
            $denom \leftarrow \alpha \sum_{i=1}^{N} \log(\frac{1}{y_i})(\frac{1}{u_i + r_i})^2(\frac{y_i}{1 - y_i})^2 G_i$
            comment: Calculate Learning Rates
            for $i \leftarrow 1$ to $num_{segment}$
              do
                $R \leftarrow (\mathbf{x} - \mathbf{v}_i)$
                $\Delta\mathbf{v}_i \leftarrow -(\frac{y_i}{1 - y_i})(\frac{u_i}{u_i + r_i})(\frac{-2\mathbf{A}_i R}{denom})$
                $\Delta\mathbf{A}_i \leftarrow (\frac{-y_i}{1 - y_i})(\frac{u_i}{u_i + r_i})(\frac{2\mathbf{R}\mathbf{R}^T - diag(\mathbf{R}\mathbf{R}^T)}{denom})$
            comment: Updation rule of $\mathbf{v}$&$\mathbf{A}$
            $\mathbf{v}_i \leftarrow \mathbf{v}_i + \sqrt{K}\Delta\mathbf{v}_i$
            $\mathbf{A}_i \leftarrow \mathbf{A}_i + \frac{c}{\sqrt{K}}\Delta\mathbf{A}_i$
            $testE \leftarrow testE + E$
        $testE \leftarrow testE/count$
        $test_f \leftarrow testE$
        $testE \leftarrow 0$
        if $(count_c = 0)\&(\frac{\lambda}{1 + 0.01\tau} > 10)$
          then $\lambda \leftarrow \frac{\lambda}{log(1 + \kappa\tau)}$
        $count_c \leftarrow rem(\tau, 15num_{segment})$
if $Check4AnySegmentLeft() = TRUE$
  then
    $\mathbf{X} \leftarrow RemovePointsNearDetectedSegments(\mathbf{X})$
    $num_{segment} \leftarrow 1$
    $RunNetwork(\mathbf{X}, num_{segment})$
```

Algorithm A.0.4: CONVERGENCE CONDITION OF NETWORK(η_{er}

```
procedure CONVERGENCECONDITION($test_f, k, array, \lambda$)
 $LimitOfArray \leftarrow 15$
 $ArrayFillFlag \leftarrow 0$
 if $\lambda > 20$
   then $\epsilon \leftarrow 10^{-8}$
   else $\epsilon \leftarrow 10^{-6}$
 if ($k < LimitOfArray$)
   then $array(k) \leftarrow test_f$
   else { $ArrayFillFlag \leftarrow 1$
          $k \leftarrow 1 + rem(k - 1, LimitOfArray)$
          $array(k) \leftarrow test_f$
 if ($ArrayFillFlag$)
   then { $s \leftarrow sum(array)$
          $eps \leftarrow test_f - s/LimitOfArray$
          if ($abs(eps) > \epsilon$)
            then $y \leftarrow array$
            else $y \leftarrow \varnothing$
   else $y \leftarrow array$
```

Algorithm A.0.5: RHT(x)

```
while (Output = $\varnothing$)
do {
    Randomly choose three pixels from the Image
    comment: Obtain estimates of tangent at each pixel
    for $i = 1$ to 3
        do {
            W$\leftarrow$A small neighbourhood (say $5\times5$)
             around the pixel $x_i$
            /*Find the line of best fit to those pixels within
             the neighbourhood (using least squares method)*/
            $m_i \leftarrow LeastSquare(\mathbf{x}_i)$
            $tangent(x_i) \leftarrow (m_i, c_i)$
        }
    comment: Finding the center
    for first with a=(1 2)& then with a=(2 3)
        do {
            Take two points on an ellipse $x_{a_1}, x_{a_2}$
            $M \leftarrow \frac{(x_{a_1}+x_{a_2})}{2}$ /* Finding midpoint */
            $T \leftarrow$intersection of $tangent(a_1)$ & $tangent(a_2)$
            comment: The ellipse center $(v_1, v_2)$ will lie on
            line $\overline{TM}$.
            /*Since transformation mapping a circle onto
            ellipse is affine, this maps lines to lines, tangents to
            tangents & intersections to intersections */
        }
    Intersection of the two lines gives an estimate of the
    center of the ellipse.
    comment: Finding the remaining three parameters
    Translate the ellipse on the origin
    This reduce eqn(1) to $ax^2 + 2hxy + cy^2 = 1$
    Substituting in the coordinates in the three pixels
    $X_1 = (x_1, y_1), X_2 = (x_2, y_2), X_3 = (x_3, y_3)$ and solving this
     for a,b,c get the remaining 3 parameters of ellipse.
    if $(ac - b^2 > 0)$
        {
        comment: The parameters represent a valid ellipse
        output $(a, b, c, v_1, v_2)$
        }
    else {
        /* Either 3 pixels do not lie on the same ellipse,
         or the estimates of tangent were in accurate.
        In either case the parameters are discarded */
        }
}
```

Algorithm A.0.6: AFCSU($\mathbf{X}, c$)

comment: Detects Ellpses in image using AFCS-U algorithm

Fuzzy c-means algorithm:
1. Initialise U_0, choose c, m, ϵ.
2. Compute all c- cluster centers $v_{i,0}$.
3. Compute all $c \times n$ membership $v_{ik,i}$ and update all c cluster centers $v_{i,i+1}$.
4. Compute E_t.
5. If $E_t < \epsilon$ then stop, else return to 3.

Algorithm 1:
1. Fix the number of clusters c, $2 \leq c < n$, where n is the number of data points. Fix $m = 1$ for hard memberships or $1 < m < \infty$ for fuzzy memberships.
2. Set iteration counter $j = 0$. Initialize the c-partition U^j.
3. Calculate $\mathbf{v}_i$ and r_i by solving the nonliner systems of equations

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (\mathbf{x}_k - \mathbf{v}_i) = 0 \tag{A.1}$$

$$\sum_{k=1}^{n} (u_{ik})^m D_{ik} = 0 \tag{A.2}$$

with d_{ik} from

$$(d_{ik})^2 = (\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x}_k - \mathbf{v}_i) \tag{A.3}$$

and D_{ik} from

$$(D_{ik})^2 = ([(\mathbf{x}_k - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x}_k - \mathbf{v}_i)]^{1/2} - r_i)^2 \tag{A.4}$$

4. Calculate $(D_{ik})^2$ as in (4).
5. Calculate S_{sfi} (or S_{si}) using

$$S_{si} = \sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T \tag{A.5}$$

and $\mathbf{A}_i$ using

$$det(\mathbf{A}_i^*) = [\rho_i det(S_{si})]^{(1/p)} (S_{si})^{-1}, 1 \leq i \leq c \tag{A.6}$$

6. Update the membership U^j, at the j^{th} iteration, using for fuzzy memberships,

$$I_k = \phi \Rightarrow u_{ik} = \frac{1}{\sum_{j=1}^{n} \left[\frac{(D_{ik})^2}{(D_{jk})^2}\right]^{\frac{1}{m-1}}} \tag{A.7}$$

or, $I_k \neq \phi \Rightarrow u_{ik} = 0, \forall i \in \overline{I}_k$ and

$$\sum_{i=1}^{c} u_{ik} = 1, \forall i \in I_k \tag{A.8}$$

For hard memberships,

$$u_{ik} = 0, \forall i \neq j, u_{jk} = 1, j \ni d_{jk} = \min_{i=1}^{c}(d_{ik}) \tag{A.9}$$

7. If converged, stop. Otherwise set $j = j + 1$ and goto step 3.

Algorithm 2:

1. Fix the number of clusters c, $2 \leq c < n$, where n is the number of data points. Fix $m = 1$ for hard memberships or $1 < m < \infty$ for fuzzy memberships.
2. Set iteration counter $j = 0$. Initialize the c-partition U^j.
3. Calculate $\mathbf{A}_i$, $\mathbf{v}_i$ and r_i by solving the nonliner systems of equations

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (\mathbf{x}_k - \mathbf{v}_i) = \mathbf{0} \tag{A.10}$$

$$\sum_{k=1}^{n} (u_{ik})^m D_{ik} = 0 \tag{A.11}$$

$$\sum_{k=1}^{n} (u_{ik})^m \frac{D_{ik}}{d_{ik}} (\mathbf{x}_k - \mathbf{v}_i)^T = \mathbf{0} \tag{A.12}$$

4. Update the membership U^j, at the j^{th} iteration, using (A.7),(A.8) and (A.9).
5. If converged, stop. Otherwise set $j = j + 1$ and goto step 3.

AFCS-U Algorithm:

1. Fix the number of clusters c, $2 \leq c < n$, where n is the number of data points. Fix $m = 1$ for hard memberships or $1 < m < \infty$ for fuzzy memberships.

2. Set iteration counter $j = 0$. Initialize the c-partition U^j.
3. Apply Algorithm 1 until convergence is achieved.
4. Use u_{ik} and $\mathbf{A}_i$ (scaled down using r_i)from step 3 as an initial guess for u_{ik} and $\mathbf{A}_i$ for the following step.
5. Calculate $\mathbf{A}_i$ and $\mathbf{v}_i$ by solving the nonliner systems of equations

$$\sum_{k=1}^{n}(u_{ik})^m \frac{D_{ik}}{d_{ik}}(\mathbf{x}_k - \mathbf{v}_i) = \mathbf{0} \tag{A.13}$$

$$\sum_{k=1}^{n}(u_{ik})^m \frac{D_{ik}}{d_{ik}}(\mathbf{x}_k - \mathbf{v}_i)(\mathbf{x}_k - \mathbf{v}_i)^T = \mathbf{0} \tag{A.14}$$

6. Update the membership U^j, at the j^{th} iteration, using (A.7),(A.8) or (A.9).
7. If converged, stop. Otherwise set $j = j + 1$ and goto step 5.

Bibliography

[1] J. Basak and S.K. Pal, *'Hough transform network'*, Electronic letters, April'95, Vol 35, No 7.

[2] J.Basak,*'Learning Hough Transform'*, Neural Computation:accepted.

[3] Rajesh N. Dave & Kurra Bhaswan, *'Adaptive fuzzy c-shell clustering and detection of ellipses'*, IEEE transaction on Neural Networks, vol 3, No-5, september 1992.

[4] P. V. C. Hough, *'Method and means for recognizing complex patterns'*, U.S. Patent 3069654, 1962.

[5] Ballard, D.& Brown, C.:*'Computer Vision'* (Prentice Hall Inc., Englewood Cliffs, NJ,1982).

[6] T. Kohonen, *'Self Organization and associative memory'*, Springer-Verlag, Berlin(1984).

[7] L. Xu & E. Oja. *'Randomized Hough transform (RHT): Basic mechanisms, algorithms, and computational complexities'*, CVGIP: Image Understanding, 57, 1993, pp. 131-154.

[8] Clark F Olson, *'Constraint Hough Transforms for curve detection'*, Computer vision and image understanding, vol. 73, no-3, March, pp-329-345, 1999.

[9] N. Kiryati & A. M. Bruckstein, *'Antialiasing the Hough transform'*, CVGIP:Graphical Models Image Process vol. 53, 1991, 213-222.

[10] N. Kiryati, Y. Eldar & A. M. Bruckstein, *'A probabilistic Hough transform'*, PR vol 24. no 4, 1991, 303-316.

[11] V. F. Leavers, *'The dynamic generalized Hough transform: Its relationship to the probabilistic Hough transforms and an application to the concurrent detection of circles and ellipses'*, CVGIP: Image Understanding 56, 1992, 381-398.

[12] V. F. Leavers, *'Which Hough transform?'*, CVGIP: Image Understanding 58, 1993,pp. 250-264.

[13] H. Li, M. A. Lavin & R. J. Le Master, *'Fast Hough transform: A hierarchical approach'*, CVGIP, 36, 1986, pp. 139-161.

[14] P. Liang, *'A new and efficient transform for curve detection'*, J. Robotic Systems 8(6), 1991, pp. 841-847.

[15] K. Murakami, H. Koshimizu & K. Hasegawa, *' On the new Hough algorithms without two-dimensional array for parameter space to detect a set of straight lines'*, in Proceedings of the IAPR International Conference on Pattern Recognition. 1986, pp. 831-833.

[16] R. O. Duda and P. E. Hart, *'Use of the Hough transformation to detect lines and curves in pictures'*, Commun. Assoc. Comput. Mach. 15, 1972, pp. 11-15.

[17] J. R. Bergen and H. Shvaytser, *'A probabilistic algorithm for computing Hough transforms'*, J. Algorithms 12, 1991, pp. 639-656.

[18] Simon X. Yang, Max Meng, *'An efficient neural network approach to dynamic robot motion planning'*, Neural Networks 13 (2000) pp. 143-148 PERGAMON.

[19] Wee Kheng Leow & Rudy Setiono, *'Explanation of the virtual input phenomenon'* Neural Networks, vol. 12 (1999), pp. 191-192 PERGAMON.

[20] N. Aras, B. J. Oommen, I.K. Altinel, *'The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem'*, Neural Networks, 12 (1999), pp.1273-1284 .

[21] Richard P.Lippmann, *'An introduction to computing with Neural Nets'*. IEEE ASSP magazine, vol 4, pp. 4-22, 1987.

[22] S. D. Shapiro, *'Generalization of the Hough transform for curve detection in noisy digital images'*, in Proceedings of the International Joint Conference on Pattern Recognition, 1978, pp. 710-714.

[23] C. Galambos,J. Kittler and J. Matas, *'Progressive Probabilistic Hough Transform for Line Detection'*, Proceedings of the CVPR(1998).

[24] Kazuhide Sugawara (Tokyo Research Laboratory, IBM, Japan) *'Weighted Hough Transform on a Gridded Image Plane'* , Proceedings of the 4th International Conference Document Analysis and Recognition (ICDAR '97).

[25] R.Soodamani & Z.Q.Liu *'Shape Analysis and Synthesis - A Fuzzy Hough Transform Approach'*, IEEE Computer Society's Student Newsletter, 1997.

[26] C. Koch, J. Marroquin, A.Yuille, *'Analog neuronal networks in early visio'*, Proceedings of the national academy of science(USA),83:4263-4267,1986.

[27] P. L. Palmer, M. Petrou, and J. Kittler, A Hough transform algorithm with a 2d hypothesis testing kernel, CVGIP: Image Understanding 58, 1993, 221234.

[28] J. Illingworth & J. Kittler,*'A survey of the Hough Transform'*, vol 44. CVGIP, 1988, pp. 87-110.

[29] M.F. Costabile & C.G. Pieroni, *'Detecting Shape Correspondence by using the generalised HT'*, 8th IJPCR, Paris, pp. 589-591.

[30] F. Evans, *'A Survey and Comparison of the Hough Transform'*, in Proc. IEEE Comp. Soc.Workshop Computer Architecture for Pattern Analysis and Image Database Management, pp.378-380, 1985.

[31] R.A. McLaughin, *'Randomised Hough Transform: Improved ellipse detection with comparison'*,

`http://ciips.ee.uwa.edu.au/Papers/Journal_Papers/`

[32] T. Risse, Hough transform for line recognition: complexity of evidence accumulation and cluster detection, Computer Vision Graphics and Image Processing 46 (1989) 327345.

[33] D.B. Shu, C.C. Li, J.F. Mancuso, Y. N. Sun, *'A line extraction method for automated SEM, inspection of VLSI resist'*, IEEE T-PAMI, Vol 10, No-1, pp. 117-120.

[34] T.C. Henderson & W.S. Fai, *'The 3D Hough Shape Transform'*,PRL 2, pp. 235-238.

[35] N. Papamarkos, *'Color reduction using local features and a SOFM neural network'*, International Journal of Imaging Systems and Technology (10) (1999) 404409.

[36] Tina Yu Tian & Mubarak Shah, *'Recovering 3D Motion of Multiple Objects Using Adaptive Hough Transform'*, IEEE-TPAMI, Vol. 19, No. 10, October 1997.

[37] R.M. Inigo, E.S. Mcvey, B.J. Bergeer, M.J. Wirtz, *'Machine vision applied to vehicle guidance'*, IEEE-TPAMI, 1984, no 6, pp. 820-826.

[38] A. Llebaria & P. Lamy, 1999, ASP Conf. Ser., Vol. 172, Astronomical Data Analysis Software and Systems VIII, eds. D. M. Mehringer, R. L. Plante, & D. A. Roberts (San Francisco: ASP), 46 .

[39] T. Shibata & W. Frei, *'Hough transform for target detection in infrared imagery'*, SPIE, no 281, 1981, pp. 105-109.

[40] P.Ballester, *'Finding Echelle Orders by Hough Transform'*, Proceedings of the ESO-ST/ECF Data Analysis Workshop, 1991.

[41] P.J. Rousseeuw & A.M. Leroy, *'Robust Regression and Outlier Detection'*, John Wiley & Sons Inc, 1987.

[42] M. Kushnir, K. Abe & K. Matsumoto, *'An application of the Hough transform to the recognition of printed Hebrew Characters'*, PR-16, 1983, pp. 183-191.

[43] W.C. Lin & R.C. Dubes, *'A review of ridge counting in dermatoglyphics'*, PR-16, 1983, pp. 1-8.

[44] S. Kasif, L. Kitchen, A Rosenfield, *'A Hough transform technique for subgraph isomorphism'*, PRL, 1983, no 2, pp. 83-88.

[45] S.D. Shapiro, *'Use of Hough transform for image data compression'*, PR 12, 1980, pp. 333-337, 1980.

[46] T. M. Breuel, *'Finding lines under bounded error'*, PR. 29(1), 1996, pp.167-178.

[47] A. Califano and R. M. Bolle, *' The multiple window parameter transform'*, IEEE T-PAMI, vol. 14, no 12, 1992, pp. 1157-1170.

The following abbreviations are used in this list:

T-PAMI	Transactions on Pattern Analysis and Machine Intelligence.
T-COMP	Transactions on computers.
T-ASSP	Transactions on Pattern Analysis and Machine Intelligence.
CGIP	Computer graphics and image processing.
CVPR	Computer vision and pattern recognition.
PR	Pattern Recognition.
PRL	Pattern Recognition letters.
PRIP	Pattern Recognition and Image Processing.
IJCPR	International Joint Conference on Artificial Intelligence.
SPIE	Society of Photogrametric and Instrumentation Engineers.